

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое
в жизни,
науке,
технике

Подписная
научно-
популярная
серия

Издается
ежемесячно
с 1988 г.

HANDHELD (о самых малых)

1991

2

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

2/1991

Издается
ежемесячно
с 1988 г.

HAND HELD

(О самых малых)

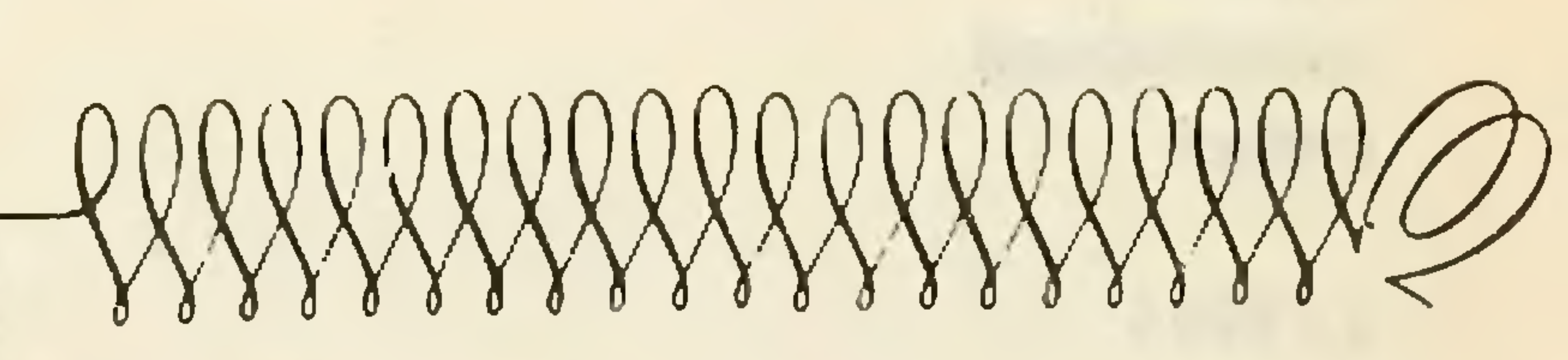
В номере:



"Знание"
Москва
1991

ББК 32.85
Х 11

Авторы выпуска



ШТЕРНБЕРГ Леонид Фрицевич — кандидат физико-математических наук, доцент Куйбышевского авиационного института.

ТАРАСЕНКО Борис Алексеевич — научный сотрудник НПО "Энергия", лектор Всесоюзного общества "Знание".

ЧАСТИКОВ Аркадий Петрович — кандидат технических наук, доцент, специализируется в области информатики и вычиспительной техники. Научные интересы автора — история, современное состояние и закономерности развития вычислительной техники.

МАЛЫХИНА Мария Петровна — кандидат технических наук, доцент, программист.

ПЕРЕХОД Игорь Александрович — кандидат физико-математических наук, доцент Симферопольского Государственного университета.

КАСАТКИН Валентин Николаевич — кандидат педагогических наук, доцент Симферопольского Государственного университета. Имеет более 100 печатных работ, из них более 20 книг, изданных в СССР и за рубежом.

РЕДАКТОР Б. М. ВАСИЛЬЕВ

ISBN 5-07-001715-2



Эта статья предназначена в помощь школьникам старших классов, изучающим информатику, и учителям, ее преподающим. А также всем, кому удалось приобрести программируемый калькулятор, и кто уже приступил к самостоятельному изучению работы на калькуляторе и изучению основ программирования.

Л.Ф.Штернберг

ПРОГРАММИРУЙТЕ С НАМИ

Урок первый

- Вадик! Мне сегодня нужна твоя помощь, — с порога обратился студент Андрей к брату-семикласснику, — мне уходить вечером, а еще чертить надо и "расчетка": считать не пересчитать.

- А калькулятор свой новенький дашь?

- Конечно, дам. И работать на нем научу.

- Тогда с удовольствием. — Вадим достал программируемый микрокалькулятор "МК-54". — Мы уже на инженерных калькуляторах считали, а на твоем я знак " = " найти не могу.

- А его там и нет: он не нужен.

- А как же считать?

- Очень просто. Числа ты набирать умеешь. Набираешь число — оно загорается на индикаторе — это число заносится в так называемый регистр X. Теперь нажимаешь клавишу "В ↑" — это число переписывается в регистр У. Набираешь второе число — оно записывается в X. Теперь нажимаешь клавишу операции " + ", или "-", или "x", или "÷" — и операция выполняется: первое число берется из У, второе — из X, результат виден на индикаторе и содержится в X. Если теперь набирать новое число, то оно появляется на индикаторе, а результат старой операции уходит в У — очень удобно выполнять цепочечные вычисления. Понял?

- Да. Давай, что вычислять?

- Подожди. Есть еще одна полезная особенность. На ваших инженерных калькуляторах есть одна ячейка памяти. А здесь их 14. Если нажать клавишу "х-П" и клавишу с цифрой, то содержимое X будет запомнено в регистре памяти с заданным номером (пока ограничимся 10 регистрами). Если нажать "П-х" и клавишу с цифрой, то запомненное число из этого регистра прочитается в X, а то, что было в X, уйдет в У. Ясно? Ну, начали. Вот таблица с исходными данными. Сюда будешь писать результаты. Действуй по моим командам.

Команды Андрея	Действия Вадима
- Бери первое число из столбца, обозначенного С.	Набирает число.
- Запомни его в регистре 0 (в Р0).	Нажимает "х-П" и "0".
- Бери число из столбца М.	Набирает число.
- Запомни его в Р1.	Нажимает "х-П" "1".
- Вызови С из Р0.	"П-х" "0"
- Где у нас сейчас М?	- В У ушло?
- Правильно. Перемножь их.	"x"
- Набери 2,5.	"2" "." "5"
- Где сейчас произведение?	- В У ушло.
- Складываем 2,5 и произведение.	" + "
- Запомни в Р3.	"х-П" "3"
- Вычисли синус от того, что в X. Это как на инженерном калькуляторе — с клавишей "F"	"F" "sin"
...	
- Все. Результат с индикатора запиши в свободный столбец.	Записывает.
- Теперь бери следующее число из столбца С и записывай его в Р0.	Набирает число, "х-П" "0"
...	

Работа кипела. Андрей чертил и командовал. Вадим работал с калькулятором. Но вскоре стало ясно, что все равно они не успевают до времени, когда Андрею надо уходить.

- Ладно, Андрей, ты иди, — сказал Вадим. — Я дальше сам управлюсь:тут все повторяется одно и то же, только с разными числами. Давай только на всякий случай запишу порядок действий, хотя я его уже и так запомнил. Итак, пишу: "Взять из колонки С, занести в Р0... Умножить на М..."

Тут в его записи заглянул отец.

- Вадик! Ну как ты пишешь?! А еще шахматист! В шахматной записи ты тоже пишешь "пойти пешкой с третьего поля во втором ряду на клетку вперед"?

- Нет, конечно. Там есть специальная краткая запись. А здесь как короче записать?

- Ну хотя бы так. — Отец взял ручку:

Занести: С,М. С·М + 2.5 → Т 3·sin T — cos T → P ... Записать: Р·Н	С — Р0 М — Р1 Т — Р2 ...

Слева записаны действия, а справа табличка: какие значения в каких регистрах хранятся.

- Хм! Действительно, здорово! Коротко и ясно. Только Т надо в Р3 хранить.

- Почему? Не все ли равно, где? Так ведь тоже можно.

В этот момент из школы пришла Аня, ученица 11 класса.

- Здравствуйте! Чем это все мужчины заняты? — она зяглянула в записи. — Вы никак тоже РАПИРой занимаетесь?

- Какой рапирой? При чем тут фехтование?

- Да не та рапира, которой фехтуют, а язык программирования РАПИРА. Мы его на уроке информатики учили. Хотя стоп: это вроде и не совсем РАПИРА, хотя очень похоже.

Ей объяснили, что к чему. Восторгу ее не было границ:

- Папа, ты же язык программирования изобрел! Почти как та РАПИРА, с которой мы работаем на машинах "Агат". Только там после каждого действия надо точки с запятой ставить, и писать "Ввод данных" вместо "Занести" и "Вывод" вместо "Записать":

ВВОД ДАННЫХ: С,М;
С·М + 2.5 → Т;
...
ВЫВОД: Р·Н;

Звонок в дверь: появился друг Андрея Виктор.

- Здравствуйте! Андрей, ты готов? Пошли. Чертеж готов у тебя?

- И еще часть расчетной работы — я помогал. — Похвалился Вадим. И он рассказал, как он считал под команду Андрея и все остальное.

Реакция Виктора была неожиданной:

- Что же ты делаешь, несчастный! У тебя же в руках **программируемый** калькулятор, а ты с ним работаешь, как с ... — Виктор не мог найти подходящего сравнения. — Выключи немедленно. Вернемся, я покажу, как с ним работать, — за полчаса все пересчитаем — и Андрея, и мое: я как раз на ваш калькулятор рассчитывал.

Пока студенты в отлучке, попробуем разобраться, что происходило. Знал ли Вадим, что он считает? Нет, не знал. (Не знаем этого и мы — формулы умышленно взяты "с потолка"). Вадим работал как **формальный исполнитель**, не задумываясь, исполнял команды. **Алгоритм** — четкую однозначную последовательность нужных действий — знал Андрей. Но дать команду "Выполни расчет" он не мог: Вадим бы его не понял. Пришлось разбить алгоритм на отдельные мелкие команды, которые Вадим мог понять (которые были в **системе команд** исполнителя) и подавать их по одной. Вадим работал с калькулятором, который также является **формальным исполнителем**: калькулятор тоже не знает, что на нем вычисляют. Вадим "переводил" команды, поданные ему, на язык, понятный калькулятору, — т.е. в последовательность нажатий клавиш. Когда потребовалось записать алгоритм, то выяснилось, что писать его на русском языке неудобно, пришлось изобрести специальную систему записи — **алгоритмический язык** (или **язык программирования** — разница между этими понятиями пока для нас несущественна). Все языки программирования, изобретенные для одинаковых или единых целей, похожи друг на друга: не удивительно, что и наш язык оказался похож на РАПИРУ. Освоивший этот

язык Вадим все равно остается формальным исполнителем — он не знает, что он считает, но теперь он более "разумный" исполнитель: теперь он понимает более сложные команды вида "С·М + 2.5 ⟶ Т". Конечно, он может их исполнить и в уме, но предпочитает переводить в последовательность нажатий клавиш.

Задача. Записать алгоритм, который по значениям диаметра и высоты цилиндра считает его объем и площадь поверхности.

Урок второй

- Ну что, продолжим? Вадик, доставай калькулятор. Показывай, Витя, как ты собрался за полчаса все пересчитать — и мое и твое.

- Смотрите, ребята. Как шла работа сначала? Андрей подавал команду, а Вадим ее тут же исполнял — он работал в **режиме интерпретации команд**, или в **режиме вычислений**. В какой-то момент Андрей дал команду "Записывай!" и стал повторять затем те же команды, что и ранее, и в том же порядке, но Вадик их уже **не выполнял, а записывал** — команда "Записывай!" как бы переключила его в **режим записи**. Затем Андрей дал команду что-то вроде "Конец записи", и Вадик отложил в сторону авторучку и снова взялся за калькулятор, т.е. **вернулся к режиму вычислений**. Теперь ему можно задать исходные данные и сказать одно слово "Вычисляй!", и он выполнит весь цикл вычислений, ничего не спрашивая у Андрея, а только поглядывая в свои записи, так сказать, **"в автоматическом режиме"**. Дальше Андрей должен задать новые исходные данные, и снова Вадик будет считать сам. Так вот, то же самое можно сделать и с калькулятором. Команда "Записывай!" подается нажатием клавиш "F" и "ПРГ" — калькулятор переходит в режим записи, который здесь называется **режим программирования**, или программный режим. Вадик, нажми.

- На индикаторе появились два нуля где-то сбоку.

- Правильно. Калькулятор может запомнить 98 команд, которые нумеруются с нулевой по 97. Номер команды еще называется ее **адресом**. Справа на индикаторе виден так называемый **счетчик адреса**, где отображен номер очередной команды. Нули означают, что сейчас будет записываться команда номер ноль. Какая была первая команда, которую ты выполнял на калькуляторе после занесения данных?

- Вызов числа из Р0: "П-x" "0".

- Нажимай эти клавиши. Что получается?

- На индикаторе "60 01". И что это такое?

- "01" означает, что сейчас будем писать команду № 1, так как команду № 0 уже записали. А "60" — это команда "взять число из Р0". Словами писать, как вы видели, долго, поэтому команды записываются **кодами** — все коды есть в инструкции к калькулятору. Что ты делал дальше?

- Умножал. Нажимаю клавишу "x". На индикаторе "12 60 02". Так: 02 — это номер следующей команды. "60", похоже, просто сдвинулось, а "12" — видимо, код команды "умножить". Да, по таблице в инструкции так и есть. Дальше надо записать результат в Р2. Нажимаю "x-П" "2": на индикаторе "42 12 60 03" — все ясно — "42" код команды "записать в Р2", остальные коды сдвинулись. А интересно, куда они дальше двигаться будут — места на индикаторе больше нет. Нажимаю "2" — на индикаторе "02 42 12 04", а "60" пропало.

- Оно не пропало. Просто сквозь индикатор видны только три последние команды, так же как в тетради видно только то, что на последней странице записано, которая еще не перевернута. Давай дальше: нажимай "5".

- Нажал: на индикаторе "05 02 42 05". Ой, мы точку нажать забыли.

- Ничего страшного. Вот клавиша "ШГ", нажми ее. Что вышло?

- На индикаторе "02 42 12 04". Ага, вроде вернулись к тому, что было раньше.

- Правильно, сделали шаг назад. Теперь нажимай то, что нужно, и правильная команда запишется в память калькулятора, стерев ошибочную.

Нажали клавиши: ".", "5", " + ", "x-П", "2", "F", "sin", "3", "x", "П-x", "2", "F", "cos", "-", "x-П", "3".

- Ладно, хватит пока. Ограничимся вычислением Р, чтоб было ясно, что к чему, а дальше сам продолжить сможешь. Заканчивать программу надо командой "С/П", а то калькулятор сам не поймет, где конец. Нажали ее: код у команды "50". Все, нажимаем "F" "АВТ", и на индикаторе снова число — можно считать дальше.

Нажмем "В/О" — это в счетчик

адреса заносится ноль, чтобы калькулятор знал, откуда начинать выполнять записанные команды. Заносим исходные данные: С = 2, М = 3, "2", "х-П", "0", "3", "х-П", "1". Клавиша "С/П" — это команда "считай по запомненным командам". Нажимаем ее — индикатор мигает несколько секунд и результат готов: 2.9974733. Аргументы синусов и косинусов, кстати, были в радианах, потому что переключатель "Р-ГРД-Г" был в положении "Р". Результат на индикаторе, но он же хранится в Р3, откуда его можно прочитать командой "П-х" "3". Теперь опять нажимаем "В/О", заносим новые данные: сколько там в таблице? С = 3,М = 6. Нажали "С/П" и через несколько секунд получаем очередной результат 3.0700535.

- Да, так действительно все за полчаса пересчитаем.

- Выключи калькулятор, включи его снова: программа в памяти стерлась. Теперь вводи команды в память снова, только теперь уже все, а не только вычисление Р, и считай.

Вадим считает. А мы еще раз вспомним нашу работу и сравним.

Разговор Андрея с Вадимом		"Разговор" Вадима с калькулятором	
Команды Андрея	**Действия Вадима**	**Команды Вадима**	**Действия калькулятора (на индикаторе)**
Взять число С	"2" "х-П" "0"	"2" "х-П" "0"	2.
Взять число М	"3" "х-П" "1"	"3" "х-П" "1"	3.
Умножь на С	"П-х" "0" "х"	"П-х" "0" "х"	6. (идет счет)
...	...	"F" "ПРГ"	...
Записывай	Берет ручку		00 ("взял ручку")
Взять число С	Пишет "С"	Эти действия будут проводиться вручную.	
Взять число М	Пишет "М"		
Умножь на С	Пишет "С·М"	"П-х" "0" "х"	Калькулятор "записывает" действия.
Добавь 2.5	...	...	
...	...	...	
Все	Кладет ручку	"F" "АВТ"	
Начнем считать	Берет калькулятор	"В/О"	Калькулятор готов
Взять число С	"2" "х-П" "0"	"2" "х-П" "0"	2.
Взять число М	"3" "х-П" "1"	"3" "х-П" "1"	3.
И считай	Выполняет все, что записано	"С/П"	Выполняет все, что записано.
Еще разок	Приготовился	"В/О"	Приготовился.
Взять число С	"3" "х-П" "0"	"3" "х-П" "0"	3.
...	...	...	...

- Витя, я кончил. Выключать, включать, и вводить твою программу?

- Не надо. Мой вариант отличается только формулой: косинус вместо синуса в одном месте и данными. Смотри: "В/О", "F" "ПРГ" — на индикаторе два нуля. А теперь нажимаем "$\overrightarrow{\text{ШГ}}$" — на индикаторе "60 01", еще раз "$\overrightarrow{\text{ШГ}}$" — "12 60 02". Что это?

- Команды, по которым мы считали.

- Правильно. Еще 5 раз "$\overrightarrow{\text{ШГ}}$": на индикаторе "1С 42 10 07" — вот и появилась команда вычисления синуса. "$\overrightarrow{\text{ШГ}}$" "F" "cos" — а вот я ее заменил на вычисление косинуса. "F" "АВТ" "В/О" — все, можешь считать с измененной программой.

Задача. Просчитайте на калькуляторе данную в прошлом уроке задачу с цилиндром для разных значений диаметра и высоты.

Урок третий

- Вадик, у меня для тебя новая работа появилась.

- Если с калькулятором, то "Всегда готов!".

- С калькулятором, с калькулятором. Только расчет похитрее, чем в прошлый раз. Смотри: Т и Н вычисляешь по этим формулам — это тебе знакомо. А дальше надо чуть-чуть думать: если Т оказалось меньше, чем T_0, то тут начинается колебательный процесс...

- Стоп-стоп. Я, как мне объяснили в прошлый раз, формальный испол-

нитель, и твой колебательный процесс, равно как и вся прочая физика, меня интересовать не должны. Ты мне объясняй только то, что я должен вычислять, а смысл результатов — это твоя забота.

- Ладно, ты прав. Итак, если T меньше T_0, то вычисляешь C по такой формуле. После чего результат вычисляется по этой формуле: обозначим ее номером 4. Если же T больше либо равно T_0, то сначала надо вычислить P — вот формула, теперь C вычисляется через P вот таким способом (*пишет формулу*). А далее результат по значению P получается по той же формуле 4. И, как в прошлый раз, все это надо вычислять для разных исходных данных. Понял?

- Понял. А интересно, как это записать "по-научному", чтобы было коротко и ясно? Вычисления по формулам ясно, как писать. Как это Аня называла — **команда присваивания**, так, что ли? А вот эти варианты: если так, то считать по-одному, а если иначе, то по-другому — это как записать?

- Не знаю. Давай спросим у папы — он что-нибудь изобретет.

- А тут и не надо ничего изобретать — откликнулся папа, слышавший весь разговор. — Запиши по-русски, так как сказал. Пиши слово "если", теперь пиши "T меньше T_0" — надеюсь, "меньше" ты не словом написал, а математическим значком? Теперь пиши "то" и формулу, по которой надо считать в этом случае.

- А формулу писать с новой строки или в той же строке?

- Как хочешь — и так и так понятно. Дальше сам управишься? Покажи, что получается.

Получилось вот что:

если $T<T_0$, то
$e^{-a\cdot T} \longrightarrow C$;
а если $T \geqslant T_0$, то
...

- Ну зачем ты второй раз выписываешь условие. Как ты говорил: "а иначе считать по-другому"? Вот так и запиши: "а" можно не писать, остается "иначе". Запятые перед "то" и "иначе" можешь тоже не писать — и так ясно.

- Как же не писать запятые? Перед "то" запятая нужна всегда.

- Если она нужна всегда, то она не нужна никогда. Вот если бы в зависимости от наличия или отсутствия запятой смысл менялся, то запятая была бы нужна. Например, в английском языке запятые перед "то" не ставят.

В результате получилось вот что:

если $T<T_0$ то
$e^{-a\cdot T} \longrightarrow C$;
иначе
$a\cdot T \longrightarrow P$;
$\sin P + \cos P \longrightarrow C$;
$C\cdot H\cdot(T-T_0) \longrightarrow R$;

- В этой записи мне ясно все, кроме одного: как я понял, по последней формуле я получаю окончательный результат и по ней я должен работать и в случае выполнения условия $T<T_0$, и в другом случае, но откуда следует, что к слову "иначе" относятся именно две формулы, а не одна и не три?

- Н-да, действительно. Ну, по логике-то понятно, что надо взять две формулы и все. А третья — это уже общая формула для обоих случаев.

- Это тебе, Андрей, понятно, потому что ты знаешь, что здесь вычисляется. А я не знаю, да и знать-то мне это как формальному исполнителю не положено.

- Тогда давай запишем эти формулы в строку после "иначе", а третью формулу выделим в отдельную строку.

- Можно, конечно, но мне это не очень нравится: Аня говорила, что присваивания можно и в строчку писать, и что главное, что их разделяет — это точка с запятой, а не новая строка.

- Тогда можно взять две формулы в скобки.

- Тоже выход. Папа, а тебе как больше нравится?

- Ни так и ни так. Как ты, Андрей, сказал — "две формулы и все"? Вот так и напиши: две формулы и слово "все", а "и" можно не писать. Как вам такой вариант нравится? А писать можно хоть в строчку, хоть в столбик: все понятно:

если $T<T_0$ то $e^{-a\cdot T} \longrightarrow C$ иначе $a\cdot T \longrightarrow P$; $\sin P + \cos P \longrightarrow C$ все $C\cdot P\cdot(T-T_0) \longrightarrow R$

И точки с запятой перед "иначе" и "все" можно не писать — и так все ясно.

- Вариант хорош. Но наши тоже годятся. Вот Аня придет — спросим у нее.

Аня подозрительно спросила:

- Папа, а ты правда никогда не занимался программированием? То, что ты предложил, — это точно то, что

есть в языке РАПИРА. Как ты так все здорово угадываешь? Вся разница, что в РАПИРЕ обычно пишут такие слова, как "если", "то" и т.д. заглавными буквами — такие слова называются **ключевыми** или **служебными**.

- Честное слово — никогда. Просто, как я понял из наших разговоров, все ваше программирование — это просто очень много здравого смысла. И если у вас есть здравый смысл, то все получится просто. А то, что я угадал столь точно, — это случайность. А ну-ка я сейчас позвоню своему приятелю — он программист — и задам ему один вопрос.

Приятель-программист подтвердил, что абсолютно все, что предлагали ребята, в языках программирования есть: и запись того, что относится к "иначе", в строку (в языке БЕЙСИК), и взятие в скобки (языки ПАСКАЛЬ, ПЛ/1, АЛГОЛ-60) — только скобки там применяют не круглые, а весьма специальные: слова "начала" и "конец", но с современных позиций вариант со словом "все" выглядит лучше.

- Ну ладно, — сказал Вадик, — с теорией разобрались. Теперь перейдем к практике: как это считать? Наверное, надо сначала вычислить Т, а затем ввести либо программу вычисления по одним формулам либо по другим.

- И сколько же раз ты будешь менять программу? Ведь нужно провести вычисления для разных исходных данных.

- А интересно, — спросила Аня, — нельзя ли ввести сразу 4 программы: вычисление Т, вычисление по формулам для случая $T<T_0$, вычисление для второго случая и вычисление итогового результата, а затем запускать их в нужном порядке?



Язык программирования ПРОЛОГ (PROLOG) создан во Франции в Марсельском университете в 1971 году. Его разработал Ален Кольмеро (Colmerauer — его фамилию по-разному переводят на русский язык).

Название языку, как говорят, дал коллега создателя — Ф.Руссель или жена последнего, хотя в дальнейшем его стали расшифровывать как "Программирование в терминах ЛОГики" (PROLOG — PROgramming in LOGic). Язык ПРОЛОГ имел непосредственного предшественника.Вот что говорит А.Кольмеро: "Риск, присущий данному проекту (проекту создания ПРОЛОГа), заключался в том, что мы могли создать язык высокого уровня, который оказался бы неэффективным. Но я однажды уже рискнул и одержал победу — так несколько лет назад был создан язык SYSTEM-Q."

Разрабатывая новый язык программирования для задач анализа и понимания естественных языков, А.Кольмеро решил использовать язык формальной логики, а точнее логики предикатов первого порядка (здесь ПРОЛОГ имеет общие корни с реляционными базами данных) и метод автоматического доказательства теорем.

Язык ПРОЛОГ основан на концепциях логического программирования, предложенных профессором Лондонского университета Робертом Ковальским, и представляет собой среду, ориентированную на рассуждения или дедукцию. "ПРОЛОГ часто путают с логическим программированием — пишет Р.Ковальский. ПРОЛОГ базируется на логическом программировании почти в таком же смысле, в каком ЛИСП базируется на ламбда исчислении". То есть можно сказать, что первым применением логического программирования является язык ПРОЛОГ. При логическом программировании, которое реализуется на языке ПРОЛОГ, осуществляется обработка символьной информации (решение проблем), в ходе которой процесс доказательства теорем интерпретируется как процесс вычисления или процесс выполнения программы. Соответствие процесса доказательства теорем и процесса выполнения программы и было установлено Р.Ковальским. А метод автоматического доказательства теорем, называемый методом резолюций, разработал Дж. Робинсон в середине 60-х годов.

Надо сказать, что попытка реализовать некоторые идеи логического программирования были сделаны при разработке малоизвестных языков QA3 и QA4, а также Хьюиттом и Зусманом из Массачусетсского технологического института при создании языков ПЛЭННЕР (PLANNER) и МИКРО-ПЛЭННЕР (MICRO-PLANNER).

До начала 80-х годов ПРОЛОГ был уделом ученых, работавших в основном в области логического программирования и мало знаком широкому кругу программистов и пользователей компьютеров (для многих непосвященных название языка ассоциировалось с аналогичным названием фирмы, производящей микропроцессорную технику).

Ситуация стала меняться с октября 1981 года, когда Япония известила мир о программе создания вычисли-

- А еще интереснее, — сказал папа, — нельзя ли выбор варианта хода вычисления поручить самому калькулятору? Это было бы гораздо удобнее.

- Придет Виктор, у него и спросим. Все равно калькулятор у него, — завершил разговор Андрей.

Задача. Написать алгорим, который по значениям *а* и *в* вычисляет max {*a*, *b*} и min {*a*, *b*}.

Урок четвертый

- Вы интересуетесь, не может ли калькулятор сам выбрать вариант работы? — спросил Виктор. — Может. Для этого у него есть 4 команды, которые называются командами **условного перехода** (хотя правильнее было бы назвать командами **условного продолжения**): это команды "F X = 0", "F X ≠ 0", "F X ⩾ 0" и "F X < 0", и команда **безусловного перехода** "БП". Эти команды хитрые — вы таких еще не видели: они занимают целых 2 ячейки в программной памяти — сама команда занимает первую ячейку, а во второй находится адрес какой-то другой команды. И заносятся они в память в 3-4 нажатия клавиши: первые 1-2 нажатия сама команда, а затем нажимаем две цифровые клавиши и заносим адрес. Вот смотрите: включаем калькулятор и нажимаем "F ПРГ". Теперь пусть надо записать команду "БП" с адресом 23: нажимаем "БП" — появился код 51, теперь нажимаем "2" — ничего не появилось, нажимаем "3" — на индикаторе появился код "23". Ясно?

- Пока не очень. Зачем это все нужно?

- Сейчас увидите. Обычно команды калькулятор выполняет подряд. А



ет принципы исполнения программы, написанной на ПРОЛОГе.

тельных машин пятого поколения, в которой язык ПРОЛОГ выбран в качестве базового языка программирования. На протяжении 80-х годов ПРОЛОГ не сходит со страниц многочисленных научных журналов по вычислительной технике и публикаций в области искусственного интеллекта.

Однако растущую популярность ПРОЛОГа, как отмечают венгерские специалисты Б.Доменки и П.Середи, нельзя объяснять только выбором авторов японского проекта, здесь кроются более глубокие причины.

1. При решении проблем "кризиса программного обеспечения", ПРОЛОГ можно рассматривать как универсальный язык сверхвысокого уровня с хорошо определенной и логически обоснованной семантикой.

ПРОЛОГ, будучи языком логического программирования, не только изменяет стиль программирования задач (в смысле фон Неймана), но и укоренившуюся (в течение четырех поколений) архитектуру вычислительных машин. "Компьютеры, по существу, никогда не менялись, они только становились меньше и мощнее, а языки приспосабливались к первоначальной архитектуре, — замечает А.Кольмеро. — Когда мы писали ПРОЛОГ, то буквально поддразнивали машину, заставляя ее делать все, что нам хотелось".

Программа на языке ПРОЛОГ содержит две составные части: факты и правила. Факты представляют собой данные, с которыми оперирует программа, а совокупность фактов составляет базу данных ПРОЛОГа, которая является не чем иным, как реляционной базой. Основная операция, выполняемая над данными, — это операция сопоставления, называемая также операцией унификации или согласования. Правила состоят из заголовка и подцелей. Выполнение программы, написанной на ПРОЛОГе начинается с запроса и состоит в доказательстве истинности некоторого логического утверждения в рамках заданной совокупности фактов и правил. Алгоритм этого доказательства (алгоритм логического вывода) и определяет принципы исполнения программы, написанной на ПРОЛОГе.

В отличие от программ, составленных на языках процедурного типа (БЕЙСИК, ПАСКАЛЬ и др.), предписывающих последовательность шагов, которые должен выполнить компьютер для решения задачи, в ПРОЛОГе программист описывает факты, правила, отношения между ними, а также запросы по проблеме.

Первый интерпретатор ПРОЛОГа, написанный на ФОРТРАНе, разработан А.Кольмеро и Ф.Русселем в 1973 году. Затем появился компилятор ПРОЛОГа, созданный Д.Уорреном и Ф.Перейра из Эдинбургского университета для машины DEC-10.

Большие успехи в развитии и распространении ПРОЛОГа были достигнуты в Венгрии, там ученые начали заниматься реализацией языка с 1974 года. В 1975 году они создали интерпретатор ПРОЛОГа, написанный на языке CDL. Основные работы по совершенствованию языка начиная с 1978 года велись в Будапештском институте по координации вычислительной техники, где в 1981-1982 годах создана модульная система ПРОЛОГ — М-ПРОЛОГ на языке CDL-2. Там же в 1982 году разработан Т-ПРОЛОГ для моделирования дискретных систем, а в 1985 году появились мини-ПРО-

нам нужно пропустить либо одну, либо другую часть команд. Вот эти команды переходов и помогут нам в этом. Команда "БП" делает вот что: после ее выполнения калькулятор будет выполнять не следующую команду, а ту, которая записана в ячейке с заданным командой "БП" адресом. В нашем случае после команды "БП 23", расположенной в ячейках ноль и один, будет выполняться команда из ячейки с адресом 23. Говорят, что "БП" выполняет **переход на адрес 23**. А команды условных переходов работают так: проверяется содержимое регистра X, если оно удовлетворяет тому условию, которое записано в названии команды, то далее выполняется следующая команда, а если нет — то происходит переход к заданному в команде адресу. Например, команда "F X=0 33" будет сравнивать содержимое X с нулем и переход к адресу 33 будет выполняться, если содержимое X не равно нулю. Ясно?

- Само по себе то, что ты рассказал, ясно. Но не ясно, что с этим делать.

- А сейчас и это поймете. Итак, нам нужно реализовать такой алгоритм:

если $T<T_0$ то $e^{-a \cdot T}$ —> C иначе $a \cdot T$ —> P; $\sin P + \cos P$ —> C все; $C \cdot H \cdot (T-T_0)$ —> R.

Предположим, что наши переменные находятся в таких регистрах: T — в P0, T_0 — в P1, a — в P2, P — в P3, C — в P4, H — в P5, R — в P6. Перед этими вычислениями нам нужно вычислить T, это мы записывать не стали, так как это мы делать умеем. Вычисление T займет у нас... сейчас напишем... ага, 10 команд — с адреса 00 по адрес 09. С

ЛОГ для персональных компьютеров и T/TC-ПРОЛОГ для моделирования дискретно-непрерывных систем. Версии, разработанные Будапештским институтом, реализованы на многих компьютерах различных фирм (IBM, Siemens, Tektronix, Motorola).

В начале 80-х годов фирма Silogic Inc. (Лос-Анджелес) создала набор интерпретаторов для разных типов компьютеров — от машин на базе микропроцессора Z 80 с ОС CP/M до машин DEC-20 с ОС TOPS-20, а также несколько компиляторов.

В последние годы наибольшую популярность получили версии ПРОЛОГа, реализованные на персональных компьютерах — сейчас в мире их насчитывается около пятнадцати. Среди них такие, как Эрити/ПРОЛОГ (Arity/PROLOG), Микро-ПРОЛОГ, ПРОЛОГ-86, ПРОЛОГ-В (PROLOG-V), Пролог-2, АЛС ПРОЛОГ (ALS PROLOG), Квинтус ПРОЛОГ (Quintus Prolog), Турбо-ПРОЛОГ и др. Только три из перечисленных — ПРОЛОГ-2, Эрити/ПРОЛОГ и Турбо-ПРОЛОГ имеют возможность компиляции. ПРОЛОГ-2 фирмы Expert Systems International (ESI) превосходит в скорости интерпретируемый ПРОЛОГ-1 в 20 раз. Эрити/ПРОЛОГ фирмы Arity (отделение фирмы Lotus Advanced Development Group) представляет собой программную среду, включающую все необходимые средства разработки программ на ПРОЛОГе (последняя версия 5.0). Фирма Borland International в 1985 году выпустила первую версию компилятора Турбо-ПРОЛОГ, которая обеспечивает многооконный интерфейс и графические возможности. (К настоящему моменту используется версия 2.0.)

На сегодняшний день не существует стандарта на язык программирования ПРОЛОГ. В то же время считается, что версия двух профессоров Эдинбургского университета У.Клоксина и К.Меллиша может быть принята за стандартную [2].

Первая аппаратная реализация языка относится к 1984 году, когда Д.Уорреном был разработан конвейерный процессор языка ПРОЛОГ — SRI. Затем в рамках реализации японского проекта машин пятого поколения в 1985 г. появился действующий макет машины PSI (Personal Sequential Inference Machine — персональная ЭВМ последовательного вывода), а в 1986 году опытный образец машины CHI (Cooperative High — Speed Inference Machine — параллельная быстродействующая машина вывода). Машина PSI с производительностью 30 тыс. логических выводов в секунду создавалась в качестве средства разработки программного обеспечения всего японского проекта. Усовершенствованный вариант PSI-II имеет производительность 150 тыс.логических выводов в секунду. Основной целью разработки CHI и ее модификации CHI-II было дальнейшее повышение производительности, которая была доведена до 400 тыс. логических выводов в секунду.

В СССР работы по логическому программированию ведутся с 60-х годов. В частности, в ЛОМИ им. Стеклова велись работы по автоматическому доказательству теорем. С.Ю.Масловым в рамках этих работ был предложен обратный метод вывода — на год раньше метода резолюций. В.Б.Борщевым и М.В.Хомяковым независимо от зарубежных работ начиная с 1972 года разрабатывалась версия логического программирования. К идеям логического программирования примыкают в этот период работы А.П.Ершова и А.С.Клещева, а также работы А.М.Степанова по описанию абстрактной параллельной машины для языков логического программирования.

Впервые в нашей стране язык ПРОЛОГ (базовый входной язык системы ПРОЛОГ-ЕС) был реализован в конце 70-х годов сотрудниками Института кибернетики им.В.М.Глушкова АН УССР и Рижского политехнического института для ЕС ЭВМ с операционными системами ДОС и ОС. Для написания основ-

адреса 10 программируем "если" T с T_o калькулятор сравнить не может, поэтому запишем условие так: "если $T-T_o<0$ то...". Итак, вычисляем $T-T_o$ и ставим команду условного перехода "F X<0":

Адрес	Команда	Алгоритм	Комментарий
10	П-х 0	если	Вызываем из Р0 значение Т,
11	П-х 1	$T-T_o$	вызываем из Р1 значение T_o,
12	—		вычитаем, сейчас в Регистре X $T-T_o$.
13	F X<0	<0 то	Теперь сравниваем содержимое X с нулем.
14	адрес		

Если X отрицательное число, т.е. $T<T_o$, то далее будет выполняться команда по адресу 15, и с этого места нужно разместить вычисление того, что записано после "то", а в противном случае будет выполнен переход к заданному адресу: там надо разместить вычисление того, что записано после "иначе".

- А с какого адреса это нужно разместить?

- Пока не знаю, поэтому адрес пока оставим незаполненным и программируем "ветвь **то**":

15	П-х 2 $e^{-a \cdot T}$	Вызываем значение а,
16	П-х 0	вызываем значение Т,
17	x	множим,
18	/-/	меняем знак
19	F e^x	вычисляем экспоненту,
20	х-П 4 —> C	и засылаем результат в С.

Теперь можно приступить к программированию "ветви **иначе**". Пока все ясно?

- Я, кажется, все понял, — сказал Вадик, — и могу продолжить сам. Теперь ясно, что ветвь **иначе** можно

ных модулей системы ПРОЛОГ-ЕС использован язык ФОРТРАН-IV, а ввод-вывод написан на АССЕМБЛЕРе.

В дальнейшем появились версии: ПРОЛОГ-СМ для мини-ЭВМ СМ-4, функционирующая под управлением операционной системы ОС РВ; система программирования ПРОЛОГ-32, реализованная на ЭВМ СМ-1700 и работающая под управлением операционной системы ДЕМОС-32; система программирования Микро-ПРОЛОГ для персональных компьютеров ЕС-1840, ЕС-1841, IBM PC/XT/AT, функционирующая под управлением операционных систем АЛЬФА ДОС И MS DOS. В Институте программных систем АН СССР разработана версия ПРОЛОГа для персонального компьютера "Ямаха MSX". Интерпретаторы языка ПРОЛОГ созданы для школьных компьютеров отечественного производства — "Корвет", УКНЦ, БК-0010.

Возможности применения языка ПРОЛОГ весьма обширны. Вот что пишет по этому поводу Р.Ковальский: "Сила и эффективность языка Пролог и его реализаций оказались такими, что он нашел многочисленные и разнообразные применения. Среди наиболее известных — применение в символической математике, планировании, автоматизированном проектировании, построении компиляторов, базах данных, обработке текстов на естественных языках, машинной индукции, отладке и экспертных системах".

Наверное, самое характерное применение ПРОЛОГа — это экспертные системы (ЭС). Пролог вносит своеобразный вклад в создание ЭС, особенно для механизма логического вывода. Модели представления знаний в ЭС включают систему продукций, фреймы, семантические сети, логику предикатов. Разработанные языки представления знаний часто зависят от тех моделей представления, которые они используют. Язык ПРОЛОГ можно использовать для применения любой модели, его можно рассматривать и как воплощение модели представления знаний, и как язык представления знаний. Определение, данное Р.Ковальским:

Алгоритм = Логика +
+ Управление

подтверждает это утверждение и помогает прояснить термин "управление" в экспертных системах. В этом определении ЛОГИКА показывает, что является проблемой (областью), а УПРАВЛЕНИЕ — как эффективно использовать логику для решения проблем.

Как утверждают создатель языка и его коллеги, "ПРОЛОГу суждено сыграть одну из первых ролей в развитии информатики", и в частности, "при правильном преподавании информатики необходимо использовать такой язык, который помогал бы структурировать мышление. Сначала пользователей взращивали на АЛГОЛе-60, затем на ПАСКАЛе. Лучше их воспитывать на ЛИСПе, но еще лучше — на ПРОЛОГе".

Литература

1. Прикладная информатика: Сб. статей. — Вып.1, 2. — М.: Финансы и статистика, 1986.
2. Клоксин У., Меллиш К. Программирование на языке Пролог. — М.: Мир, 1987.
3. Логическое программирование/Пер. с англ. и фр. — М.: Мир, 1988.
4. Дедков А.Ф. Логическое программирование. — М.: Знание, 1988.
5. Мир ПК, 1988, № 2; 1990, № 2, 3.
6. Доорс Д, Рейблейк А.Р., Вадера С. Пролог — язык программирования будущего/Пер. с англ. — М.: Финансы и статистика, 1990.
7. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог/Пер. с англ. — М.: Мир, 1990.

М.Малыхина,
А.Частиков

начинать с адреса 21. Поэтому в пропущенный адрес можно записать число 21 и продолжить с адреса 21:

21	П-х 2	а·Т	Опять вызываем значение *а*,
22	...	...	и т.д.

Если окажется, что $T > T_0$, то калькулятор пропустит команды 15-20 и сразу начнет с адреса 21. Так?

- Так. Но все же ты немного ошибся. Если окажется, что $T < T_0$, то будут выполнены команды с адресами 15-20, а потом что?

- Потом калькулятор остановится.

- Интересно, почему это он вдруг остановится? Чем команда по адресу 20 отличается от команды по адресу 19? Остановиться он может только по команде "С/П". А без нее он начнет выполнять команды по адресам 21, 22 и т.д.

- Точно. Надо добавить команду "С/П", а потом программировать ветвь **иначе**.

- Идея заманчива, но... А вычислять значение R ты будешь потом вручную?

- Н-да, опять не то. Тут надо пропустить выполнение части команд, перепрыгнуть через них так же, как перепрыгиваем через ветвь **то**.

- Верно. А для этого надо поставить не "С/П, а "БП" и опять же адрес выяснится только тогда, когда кончим программировать ветвь **иначе:**

21	БП	**иначе**	Пропускаем команды ветви **иначе**
22	адрес		Адрес заполним позже
23	П-х 2	а·Т	Выполняем вычисления ветви иначе
24	П-х 0		вызываем значения *а* и Т
25			множим
...	...	...	и т.д.
31			Складываем sin P и cos P
32	х-П 4	→ C	и помещаем результат в C
		все;	Вот теперь можем в ячейку 14 записать адрес 23,а в ячейку 22 — адрес 33
33	П-х 4	C·H	И приступаем к вычислению значения R,
34	П-х 5		
...	...		после завершения которого уже можно ставить "С/П"

- Понял. Но есть вопрос: а если бы в условии стояло не "$T < T_0$", а "$T \leq T_0$", что делать? Ведь команды "$F\ X \leq 0$" нет.

- Тогда запишем "если $T_0 - T > 0$ то ..." и все.

- Еще вопрос: а как калькулятор отличает адрес "23" от кода "23", который имеет команда "F 1/x"?

- А как ты определяешь, что запись "09/10-87" — это 9 октября, а не 10 сентября?

- Я просто знаю, что сначала идет день, а потом месяц.

- И он "знает", что после команд переходов идет не код, а адрес.

- Ну ладно, я пошел считать.

Задача. Написать программу для калькулятора, определяющую минимум и максимум для двух чисел.

Урок пятый

Сегодня Вадик получает от Андрея очередной "заказ":

- Вадик, не хочешь мне посчитать несколько интегралов?

- ??? А думаешь, я знаю, что это такое?

- Да ты не бойся, это только название страшное, а на самом деле все просто. Дана функция f(x) и значения *а* и *в*, а также число N. Разбиваем отрезок [*а*, *в*] на N одинаковых частей, каждая длиной H = (*в*-*а*)/N. Получаем N точек: $x_1 = a$, $x_2 = a + H$, $x_3 = a + 2 \cdot H$, $x_4 + a + 3 \cdot H$ и т.д. А теперь считаем сумму $f(x_1) \cdot H + f(x_2) \cdot H + \ldots + f(x_N) \cdot H$. Эта сумма дает приближенное значение площади фигуры, ограниченной графиком функции, осью X, и прямыми *y* = *а* и *y* = *в*. Площадь фигуры называется *определенным интегралом от функции f(x) на отрезке [а, в]*. А так вычисляется его приближенное значение. Вот и все.

- И всего-то? А я-то думал: интегра-а-ал!

- А в учебнике это называется "площадь криволинейной трапеции", — сказала Аня. — Только не написано, зачем такие площади вычислять.

- Для многих целей. Например, если f(*x*) — это зависимость скорости от времени, то интеграл — это пройденный путь за время с момента *а* до момента *в*. Если f(*x*) — расход воды или топлива в момент *x*, то интеграл — это общая затрата воды или топлива с момента *а* до момента *в*. Есть еще множество задач такого типа.

- Понял. Пиши алгоритм.

- Алгоритм простой:

0 → сумма; а → x; (в-а)/N → H;
сумма + f(x)·H → сумма; x + H → x;
сумма + f(x)·H → сумма; x + H → x;
и т.д.

Ясно?

- Отнюдь. Во первых, зачем каж-

дый раз умножать на Н: можно сначала накопить сумму $f(x_i)$, а затем умножить ее на Н. Так?

- Ты прав, можно.

- А во-вторых, как я должен программировать твое "и т.д."? Сколько таких строк будет?

- N штук.

- А чему равно N?

- По-разному, где 20, а где 100.

- Так сколько же таких строк я должен написать: 20 или 100? И кстати, все равно программа в память калькулятора не поместится ни при 20, ни тем более при 100 строках. Аня, что в таких случаях делают?

- Сейчас я папу позову — мне интересно, что он придумает, хотя я знаю решение: мы это в школе изучали.

- Подожди, — решительно сказал Вадик. — Сами справимся. Итак, демонстрируется фокус, — он подтянул рукава жестом фокусника. — Сейчас я изобрету новую конструкцию алгоритмической записи. Что нам нужно? Одни и те же действия надо повторить несколько раз. Как это записать? Отвечаю: записать надо **по-русски**, пишем "повторять N раз", затем пишем, что повторять, а далее все остальное. Для красоты, как учила Аня, и удобства чтения такие специальные слова подчеркиваем. Получаем:

повторять N раз

сумма + f(x) ⟶ сумма; x + H ⟶ x
сумма ·H ⟶ сумма;

Фокус окончен. Аня, есть такая штука в РАПИРЕ?

- Конечно, есть.

- Тогда все в порядке. Не слышу аплодисментов.

- Фокус не кончен, поэтому аплодисментов нет. А откуда видно, что надо повторять два присваивания, а не три и не одно?

- Ты права. Но это мы уже проходили: добавим где надо слово "**все**".

- "**Все**" нельзя, — сказал Андрей. — Оно уже использовалось для завершения **если...то...иначе**.

- Ну тогда... — Вадик задумался, — напишем "коточнапо".

- Чего-чего напишем?

- "коточнапо": КОнец ТОго, Что НАдо ПОвторять. Чем плохо?

- Ну ты придумаешь! Ничего себе словечко! А почему нельзя "**все**"? — спросила Аня.

- Ну как же! — сказал Андрей. — А если у нас есть и "**если**" и "**повторять**", тогда как определить, какое "**все**"к чему относится? Вот так например:

повторять ... **раз**
если ... **то** ... **иначе** ...
все все

- Очевидно: первое "**все**" закрывает "**если**", а второе — относится к "**повторять**". Только для удобства чтения это пишут немножко не так, а так, чтобы ясно было, что к чему относится:

повторять...раз
 если...то...иначе...все
все

и сразу ясно, что к чему относится.

- А если я хочу, чтобы первое "**все**" закрывало "**повторять**", то надо так написать, что ли? *Пишет:*

повторять...раз
| **если...то...**
| | **иначе...**
все...|
 все

- Хотеть-то ты можешь, только что бы это значило? В какой последовательности должны выполняться отмеченные точками команды?

- Хм... действительно. В общем, ясно: так пересекаться они не могут, а могут быть только одна внутри другой либо одна вслед за другой. Аня, а что говорит по поводу "**все**" наука?

- Наука говорит, что на ЭВМ "Агат" пишут "**все**", и всегда ясно, какое "**все**" к чему относится, а в учебнике пишут "**кц**", т.е. "конец цикла" — то, что Вадик сочинил, называется *командой повторения* или *циклом*, а те команды, которые повторяются — у нас их две — *телом цикла*. Мне вообще больше "**кц**" нравится.

- Ладно, пишем "**кц**". Получаем:

0 ⟶ сумма; а ⟶ x; (в-а)/N ⟶ H;
повторять N раз
 сумма + f(x) ⟶ сумма; x + H ⟶ x
кц
сумма·H ⟶ сумма

Результат в переменной — **сумма**. Остается разобраться как это реализовать на калькуляторе. Тут есть еще не использованные нами клавиши — наверняка они пригодятся. Сейчас посмотрим в инструкции.

В инструкции обнаружили 4 команды: "F L0","F L1", "F L2" и "F L3", явно специально предназначенные для таких случаев. Каждая из них занимает 2 ячейки: сама команда и адрес. Эти команды вычитают по единице из регистра с упомянутым в ко-

манде номером (т.е. P0, P1, P2 или P3), и если в результате получился ноль, то далее выполняется следующая команда, а если нет, то происходит переход к заданному адресу. С помощью этих команд все получается просто. Пусть *сумма* — в P1, *а* — в P2, *в* — в P3, x — в P4, H — в P5, *в* N — в P6, P0 оставили для *счетчика* числа повторений.

Адрес	Команда	Алгоритм	Комментарий
00	0		Засылаем ноль в регистр, где
01	x-П 1	0→сумма;	будем накапливать сумму.
...	...		Аналогично вычисляем x и H.
10	П-x 6	повторять	Засылаем число повторений
11	x-П 0	N раз	в счетчик.
12	П-x 0	сумма +	Выполняем вычисления. В ка-
13	П-x 4	sin(x)	честве функции взяли y = sin x.
14			
...	...	...	
21	F LO	кц	Эта команда будет вычитать
22	12		по единичке из P0. После N-го вычитания получим ноль и перейдем к команде 23. А первые N-1 раз будет происходить переход на адрес 12.
23	x-П 0	сумма.H	Остается домножить сумму
24	...	...	на H.

Задача. Дополните эту программу нужными командами вместо многоточий и вычислите площадь одной дуги синусоиды, взяв N = 20. Должно получиться приблизительно 2.

Задача. Напишите алгоритм и программу для калькулятора для вычисления 1·2·3·...·(M-1)·M, где M — заданное число.

Урок шестой

- Папа, Аня, Андрей! Вот билеты на выставку, а с ними задачка на составление алгоритма. Слушайте. Занял я очередь, а она длинная-длинная, и пошел за Сашей, чтоб не скучно было. Прихожу минут через 20 — никак не могу найти, где я стоял. Что делать? Я тогда подхожу к первому в очереди и спрашиваю, давно ли он стоит. Он говорит: "Минут 50". Ага, думаю, значит, моя очередь никак пройти не могла. Тогда я подхожу к середине очереди и спрашиваю у кого-то то же самое — он говорит: "Минут 15". Ясно, что он занимал позже меня. Тогда я беру середину от первой половины — там говорят: "С полчаса". Тогда я беру середину от второй половины первой половины и т.д. В общем, когда район поиска сузился человек до 10, я увидел того, за кем занимал.

- Молодец, Вадик! — сказала Аня. — То, что ты сочинил, называется "решение уравнения методом деления пополам".

- Какого уравнения? — удивился Вадик. — Какое может быть в очереди уравнение?

- В очереди уравнения нет, — сказал Андрей, — но **математическая модель** твоих действий — это и есть решение уравнения. Время, которое человек стоит в очереди, есть функция от номера этого человека. В твоем случае f(1) = 50, т.е. первый человек стоит 50 минут, а f(M) = 0, т.е. последний, M-й человек стоит ноль минут. Тебе надо найти того, кто стоит около 20 минут, т.е. найти x такой, что f(x)≈20, т.е. решить уравнение f(x)-20 = 0, разумеется, решить приближенно. И ты с этим прекрасно справился.

- Здорово! Никогда бы не подумал, что математику можно применить к стоянию в очереди. А интересно, как записать этот алгоритм?

- Математику можно применять в очень многих случаях. И то, что в самых непохожих случаях получаются одинаковые модели. Прежде чем писать алгоритм, надо разобраться с моделью. У нас имеется функция Ф(x) такая, что Ф(a) > 0, Ф(в) < 0, и на отрезке [*а*, *в*] функция убывает. Ясно?

- Не совсем, — сказал Вадик. — Вроде, была f(x), и она всегда положительна.

- Нет, Ф(x) это не то же, что f(x). В нашем случае, Ф(x) — это время, на которое человек № x стоит в очереди дольше тебя, т.е. Ф(x) = f(x)-20.

- Вот теперь ясно.

- В общем, опуская строгую математику (на интуитивном уровне и так все ясно), можем решать это уравнение. Обычно точный ответ и не нужен, задают некоторое ε, и как только найдем корень с точностью ε, считаем, что этого достаточно. У тебя было ε = 10, т.е. тебе надо было найти нужное место плюс-минус 10 человек. Если обозначить левый конец отрезка через *лев*, а правый через *прав*, то один шаг алгоритма можно записать так:

(прав + лев)/2 → x;
если Ф(x) > 0 **то** x → лев **иначе**
x → прав **все**

т.е. на каждом шаге либо левая либо правая граница отрезка сдвигается к середине и отрезок [*лев*, *прав*] со-

кращается вдвое. Теперь достаточно организовать цикл и все.

- Андрей, а как ты организуешь цикл? В прошлой задаче было ясно, сколько раз он должен выполняться, а здесь как быть? Повторять эти шаги надо до сужения отрезка в достаточной степени, а сколько раз — это неизвестно. Я что-то не соображу, что нужно делать.

- Ну, если ты не сообразишь, спросим Аню. Или лучше папа сейчас изобретет что-нибудь новенькое. А, папа?

- А что тут изобретать? У вас был цикл с заранее известным числом повторений: **повторять** М **раз**. А теперь нужен другой цикл, где число повторения заранее не известно. Пишем его, как всегда, самым простым и обычным способом:

повторять *то, что надо повторять* **до**, *условие завершения.*

В вашем случае алгоритм принимает вид:

1 → лев; М → прав;
повторять
(прав + лев)/2 → x;
если Ф(x) > 0 **то** x → лев **иначе** x → прав **все**
до прав — лев < $\mathcal{E}$;

Когда *прав-лев* станет меньше $\mathcal{E}$, то повторение шагов закончится, а в *x* будет значение корня. Ясно?

Аня скептически посмотрела на эту запись:

- Папа, на сей раз ты изобрел что-то не то: нет такого ни в учебнике, ни в РАПИРе, с которой мы работаем на машинах "Агат".

Теперь уже удивился папа:
- Не может быть! Такая естественная запись! Где-нибудь что-то подобное обязательно есть. Впрочем, сейчас спросим у специалистов. — И он взялся за телефонную трубку.

- Все логично, как я и ожидал, — продолжал папа, закончив разговор. — Есть, конечно же, такой цикл — в языке ПАСКАЛЬ, например. А у вас в учебнике, наверное, другой вариант, тоже не менее логичный

пока *условие* **повторять**
то, что надо повторять **кц**

Так, что ли?
- Да, — сказала Аня. — Только вместо "**повторять**" мы пишем **нц**, т.е. "начало цикла", но это уже мелочи. А этот алгоритм в учебнике записан так:

1 → лев; М → прав;
пока прав-лев ≥ $\mathcal{E}$ **повторять**
(прав + лев)/2 → x;
если Ф(x) > 0 **то** лев → x **иначе** прав → x **все**
кц

- Можно и так, это одно и то же, — сказал Вадик.

- Не совсем, — сказал папа. Во втором варианте если с самого начала *прав-лев* было меньше $\mathcal{E}$, то шаг вычислений не будет выполняться ни разу. Впрочем, пока что этой разницей можно пренебречь.

- А как это сделать на калькуляторе, — спросил Вадик. — Наверное, есть специальные команды?

- Зачем? — ответил Андрей. — Команд условного перехода вполне достаточно. Запрограммируем, например, первый вариант. Пусть М будет в Р0, прав — в Р1, лев — в Р2, x — в Р3, $\mathcal{E}$ в Р4. Программа будет иметь вид:

Адреса	Команды	Комментарий
00	1	Засылаем единицу
01	x-П 2	в лев.
02	П-x 0	Пересылаем М
03	x-П 1	в прав.
04	...	Начиная отсюда программируем тело
	...	цикла, присваивания и условные команды мы программировать умеем
20	П-x 1	А теперь вычисляем прав-лев-$\mathcal{E}$.
21	П-x 2	
22	-	
23	П-x 4	
24	—	Если прав- лев-$\mathcal{E}$<0, то прав-лев<$\mathcal{E}$.
25	FX<0	Если условие не выполнено, то
26	04	возвращаемся на начало цикла и повторяем шаг вычислений
27	...	

- Понятно, — сказал Вадик. — А как вариант с **пока** сделать, я и сам соображу: все аналогично. Оказывается, иногда полезно и в очереди постоять.

Задание. Попробуйте полностью выписать программу для калькулятора для какой-либо конкретной функции и провести по ней вычисления.

Урок седьмой

Сегодня Вадим опять программирует очередной алгоритм для Андрея с Витей. ПМК уже освоен довольно неплохо, и Вадим справляется с большинством проблем сам. Но иногда возникают вопросы.

- Андрей, и где ты только такую константу могучую раздобыл — 6.02252×10^{23}?

- Хорошая константа, называется числом Авогадро. Это количество атомов в одном литре газа. А чем она тебе не нравится?

- Только тем, что для ее набора надо аж целых 10 команд, а набирать ее придется в трех местах, и в итоге у меня программа в память не лезет.

- Ну набери один раз и запиши в регистр, а потом вызывай из него.

- Регистры все заняты. А покороче ее набрать нельзя — не все десятичные цифры?

- Очень нежелательно: точность расчета потеряется.

- Что же делать? Хорошо с константой — есть для нее специальная клавиша. Жаль, на все константы клавиш не напасешься. Вот бы где-то в одном месте написать команды набора этой константы, а потом сказать ПМК: "А ну-ка, выполни их".

- Ну и скажи, в чем проблема? — спросил Виктор. — Как перейти с одного места программы в другое ты знаешь.

- Как перейти, знаю: командой "БП", а вот как назад вернуться не знаю. Переходить-то надо будет из трех разных мест, откуда ПМК будет знать, куда надо вернуться?

- Если переходить командой "БП", то, конечно, назад не вернешься, но есть еще одна команда перехода "ПП", которая отличается от "БП" только тем, что, кроме перехода, она запоминает в специальном регистре (который увидеть никак нельзя) еще и адрес следующей команды, т.е. как раз то место, куда надо вернуться. А чтобы вернуться, есть еще одна команда "В/О", которая выполняет переход по адресу, записанному в этом регистре. А теперь пусть с адреса, скажем 80, мы поместили команды набора нашей константы:

6 . 0 2 2 5 2 ВП 2 3 В/О

в конце записали команду "В/О", а теперь где-то в программе написали команду "ПП 80", скажем, по адресам 10 — 11. Что получится? После выполнения этой команды будет выполнен переход на адрес 80, а в специальном регистре будет запомнен адрес следующей команды, т.е. 12. Дальше начнут выполняться команды, записанные по адресам 80, 81 и т.д., которые занесут в регистр X так не понравившуюся тебе константу, а затем будет выполнена команда "В/О", которая выполнит переход по запомненному адресу, т.е. к ячейке 12, и можно работать дальше с занесенной в X константой.

- Здорово! Получается, что мы как-бы научили ПМК новой команде, которой у него раньше не было: ведь теперь команда "ПП 20" мало чем отличается от команды "F пи".

- Действительно так. Эта штука называется подпрограммой. Команда "ПП" расшифровывается как Переход к Подпрограмме, а "В/О" — Возврат/Обнуление (дробь в названии показывает, что эта команда поразному работает в программе и в автономном режиме). Как теперь писать программу, понятно?

- Да, программа получается такой:

Адрес	Команда	Комментарий
...	...	
10	П-х 1	Вызываем значение из P1,
11	ПП	обращаемся к подпрограмме
12	80	для занесения константы в PX
13	x	и умножаем эти значения.
...		...
23	П-х 4	Вызываем значение из P4,
24	ПП	обращаемся к подпрограмме
25	80	для занесения константы в PX,
26	x	и умножаем эти значения.
...		...
80	6	Эти 10 команд
81	.	заносят в регистр X
82	0	значение константы.
83	2	
84	2	
85	5	
86	2	
87	ВП	
88	2	
89	3	
90	В/О	И выполним переход к запомненному месту

- Все прекрасно, — сказал Вадик, — теперь программа хоть с трудом, но в память поместится.

- Я только вот что не пойму: откуда у тебя 3 раза встретилось занесение этой константы — там ведь одна и та же функция. Ты запрограммируй ее один раз и все.

- Я бы с удовольствием, но один раз эту функцию надо вычислять от аргумента a, второй раз — от b, третий — от x, а они все в разных регистрах хранятся, и результаты надо опять же в разные регистры класть. Вот, кстати, интересно: надо сделать одинаковые действия, но с разными регистрами. Жаль, что нельзя заставить ПМК это сделать, но чтоб эти действия писать один раз, а не несколько.

- Ну почему же нельзя? А как ты заставляешь команду сложения работать с разными числами, хранящимися в разных регистрах.

- Никак. Просто пересылаю нужные числа в стек, а команда сложе-

ния сама знает, что данные всегда надо брать из стека и результат оставлять там же. Стоп! Кажется, понял. Надо данные заслать в какое-то заранее оговоренное место, откуда подпрограмма их будет брать и в каком-то заранее оговоренном месте она должна оставлять результат.

- Точно. И какое-же место ты выберешь? Только его желательно выбрать так, чтобы можно было договориться раз и навсегда, и чтобы выбор был логичен. Вспомни, что мы как бы научили ПМК новой команде.

- Намек понял: надо чтобы подпрограмма работала по аналогии с командой — тогда надо исходные данные ей задавать в стеке, а результат она должна тоже оставлять в стеке.

- Абсолютно верно. Итак, у нас там функция

$$f(x) = 6.02252 \times 1023 \cdot x^2 + 1,$$

а вычислять ее надо один раз от a, второй раз от b, а третий от x, который вычисляется. Как это программировать?

- Единственное данное — это аргумент функции, его, судя по тому, что ты сказал, надо заслать в стек и перейти к подпрограмме, она должна все вычислить и оставить результат в стеке же. А после возврата с этим значением в регистре X мы можем делать, что хотим. Пусть подпрограмма размещается с того же адреса 80. Головная часть тогда должна иметь вид:

Адрес	Команда	А-язык	Комментарий
		алг ...	
	a - P0	вещ a,	
	b - P1	b,	
	...- P2	...,	
	x - P3	x	
15	П-x 0	???	Засылаем значение a в стек (в регистр X)
16	ПП		
17	80		и обращаемся к подпрограмме.
18	2		А результат можем, например,
19	:		поделить на 2.
28	П-x 1		А теперь засылаем в стек
29	ПП		значение b и опять обращаемся
30	80		к той же подпрограмме,
31	x-П ...		а результат можно куда-то .... записать
80	F x²	результат:=	Сама же подпрограмма возводит
81	6	x**2 *	аргумент в квадрат,
82	.	6.022...	заносит на регистр X эту
83	0		неудобную константу,
91	3		
92	x		множит на нее x²,
93	1	+1	и складывает с 1.
94	+		Результат остается в X.
95	В/О		И возвращаемся назад.

- Да, так конечно программа стала куда как короче и со всех сторон удобнее. А интересно, этот трюк, который мы проделали с ПМК, можно как-то записать на алгоритмическом языке? Аня, у вас в учебнике есть что-нибудь подобное?

- Тебе уже папа объяснял, что если что-то очень логично, то оно обязательно есть там, где надо. Естественно, что ситуация, когда надо один раз объяснить человеку или ПМК действия, а потом много раз их выполнять, может быть, с еще какими-то вариациями, не у нас первых возникла. Идея-то крайне проста: сначала объясняешь алгоритм действия, а потом говоришь: "Давай, выполняй!"

- Ну и как это сказать?

- Во-первых, алгоритму надо дать название; до сих пор мы с тобой писали только так называемое тело алгоритма, т.е. сами выполняемые команды, а полностью алгоритм пишут так:

алг название(арг аргументы, рез результаты)
нач
тело алгоритма
кон

- Например, для вычисления твоей функции нужно написать

алг функ(арг вещ x, рез вещ y)
нач
y: = 6.02252x1023·x² + 1
кон

- Это означает, что алгоритм называется "функ", в качестве исходных данных он получает одно значение: вещественную переменную, которую называет x, и выдает он тоже один результат, который называет y. А кроме этого алгоритма, можно написать другой алгоритм, в который включается имя первого алгоритма, и задается то, с чем он должен работать и куда положить результат. Для этого надо просто написать имя нашего алгоритма и в скобках написать конкретные числа или переменные. Например,

функ (7.3,t)

означает "выполнить алгоритм функ, взяв в качестве x значение 7.3, а результат положить в переменную t". Такой алгоритм, к которому обращаются из другого алгоритма, называется вспомогательным, а тот, который обращается, — главным. Кстати, если алгоритм еще имеет какие-то переменные, кроме аргументов и результатов, то эти переменные надо описать после слова "нач". Понял?

- Итак, у нас имеется два алгоритма: основной — наш старый знакомый — вычисление площади криволинейной трапеции, или, по-научному, интеграла, который теперь написан не для конкретной функции, а для некоторой, которая называется F, и сам алгоритм вычисления этой некоторой

функции, который показывает, как ее надо вычислять:

```
алг площадь;
нач  вещ a,b,x,сум,y,h  цел N
 ввод a,b,N;
 сум: = 0; h: = (b-a)/N; x: = a
 для i от 1 до N-1 нц
   x: = x + h; F(x,y); сум: = сум + y;
 кц;
 F(a,y); сум: = сум + y/2; F(b,y);
 сум: = сум + y/2;
 вывод сум*h
кон
алг F(арг вещ x, рез вещ y )
нач
 y: = 6.02252x1023·x^2 + 1
кон
```

- Андрей, а почему у тебя алгоритм вычисления интеграла отличается от того, который ты мне давал раньше?

- Это просто немного другой метод, он точнее результат дает, хотя формулы чуть-чуть сложнее: тот называется "метод прямоугольников", а этот "метод трапеций", но это сейчас не важно. Ну что, все проблемы решили?

- Да я бы не сказал, что все. Мне этот вспомогательный алгоритм както не нравится: какая-то запись неестественная. Папа говорил, что все, что естественно пишется, должно и писаться нормально, а тут все-таки некоторая ненормальность есть. Мы ведь формулы для накопления суммы вычисления итогового значения как писали? Математически:

сум = сум + f(x)

S = h((f(a) + f(b))/2 + s)

Хотелось бы и в алгоритме так же написать, а тут какой-то (y) во вспомогательном алгоритме, которого в формулах совсем нет. И алгоритм получился совсем не похожий на математическую запись.

- Ну, Вадик, — вмешался папа, — мы же это с тобой уже проходили: а ну-ка, изобрети запись, которая была бы тебе удобна, и я тебе гарантирую, что ты изобретешь чтото, что уже где-то наверняка есть.

- Ладно, сейчас попробую. Вопервых, я хочу, чтобы основной алгоритм писался так, как в математической записи. То есть, он должен выглядеть как-то так:

```
алг площадь;
нач  вещ a,b,x,сум,y,h  цел N
 ввод a,b,N;
 сум: = 0; h: = (b-a)/N; x: = a
 для i от 1 до N-1 нц
   x: = x + h;  сум: = сум + F(x);
 кц;
 сум: = ((F(a) + F(b))/2 + сум)*h
 вывод сум
кон
```

- Отлично! Теперь сообрази, как записать вспомогательный алгоритм.

- Записать надо как-то так же, как и было, но вот только чему присвоить результат? Так что ли записать?

```
алг F(арг вещ x )
нач
 y: = 6.02252x1023·x^2 + 1
кон
```

- Так нельзя: у тебя (y) не описан, а значит, когда начнешь писать программу для ПМК, то не будешь знать, какой регистр для него взять.

- Ладно, давай опишем *y*.

```
алг F(арг вещ x )
нач вещ y
 y: = 6.02252x1023·x^2 + 1
кон
```

- Так, конечно, можно, но при этом ты на ПМК отведешь для (y) регистр, подпрограмма зашлет туда значение функции и на сем удовлетворится. А интересно, как его оттуда достанет главная программа?

- Да, здорово получается: ты говоришь вспомогательному алгоритму "давай выполняйся", он отвечает "я завершился", а результат оставляет при себе.

- Ладно вам хихикать, как же здесь записать?

- Ты в формуле что пишешь? — спросил Андрей. — Сум плюс что?

- Плюс F от икс.

- Вот именно что F. Так чему надо присвоить результат?

- Имени алгоритма F, что ли?

- А почему бы и нет? Так и записать:

```
алг F(арг вещ x )
нач  F: = 6.02252x1023·x^2 + 1
кон
```

В итоге имя F получает значение и его, т.е. имя, можно использовать в сложении, умножении и других действиях так, как это сделано в основном алгоритме. Папа, как идея?

- Хм, вообще-то тоже ничего, хотя я, честно сказать, думал по-другому. Вадик, ты как спрашивал: чему бы присвоить значение результата? Так? Ну, так и напиши:

результат: = 6.02252x1023·x^2 + 1

Получается, что нам нужно новое служебное слово "результат", вот и все. Аня, а ну-ка, проведи экспертизу.

- Папа, у тебя просто какое-то чутье! В учебнике написано ну просто то же самое, только вместо слова

"результат" используют слово "знач" да и перед словом "алг" в этом случае зачем-то пишут слово "вещ", а слово "арг" почему-то разрешается опускать.

```
вещ алг F(вещ x )
нач знач: = 6.02252x1023·x² + 1
кон
```

И называется все это "алгоритм вычисления значения функции", хотя разницу и смысл я не совсем понимаю.

- А я, пожалуй, объясню. Такой способ организации вспомогательного алгоритма пройдет только тогда, когда алгоритм имеет только один результат, и тогда не надо писать, что для него аргументы, а что — результаты: все — аргументы. А кроме того теперь имя алгоритма F играет две роли: во-первых, оно имя алгоритма, а во-вторых, оно как бы переменная, так как получает значение функции. Именно это значение мы и используем, когда пишем

сум: = сум + F(x).

Вот поэтому мы и описываем F и как заголовок алгоритма и как простую переменную. А вариант, который Вадик предложил, тоже ничего — наверняка он где-нибудь встречается. (И действительно встречается: в языках ФОРТРАН и ПАСКАЛЬ, например.) Ну что, теперь все ясно?

- Теперь все, можно писать программу.

Задание. Напишите вспомогательный алгоритм, который вычисляет длину отрезка на плоскости по координатам (x1,y1) и (x2,y2). Напишите головной алгоритм, который вводит координаты вершин треугольника и с помощью трех обращений к вспомогательному алгоритму получает периметр треугольника. Напишите соответствующую программу для ПМК и проверьте ее работу.

Урок восьмой

- Вадик! Ну, как успехи в освоении программирования? — спросил, заходя в комнату наш главный консультант — папин сотрудник, программист.

- Ничего, уже изобрели ветвления, циклы, а вчера еще и подпрограммы изобрели.

- Молодцы! Вы уже изобрели все программирование.

- Как все?

- А так: в музыке есть семь нот, и все состоит из них, а в программировании ноты всего четыре: простое действие (вычисление, ввод, вывод), ветвление, цикл и подпрограмма — все. Больше в нем ничего нет. В музыке все ноты комбинируются всего двумя способами: последовательное звучание и одновременное — аккорд, и в программировании всего два: следование и вложение.

- Так что, мне больше изучать в программировании нечего, что ли?

- А в музыке как? Ноты освоил и больше изучать нечего? Нет, остается еще малость — научиться играть. Так что и здесь малость осталась — научиться программировать; тут совершенствоваться можно до бесконечности. Но с первой ЭВМ — калькулятором — ты уже вполне прилично познакомился.

- Владимир Михайлович, я только что-то никак не пойму: калькулятор — это ЭВМ или нет? Я тут Анин учебник читал — так там устройство ЭВМ ДВК-2 описано, ну, абсолютно не похоже на ПМК. Какая-то двоичная система счисления, биты, байты, кодировка — ничего не ясно и абсолютно все не похоже на ПМК. А вы с Витей меня уверяли, что ПМК — это ЭВМ, но только маленькая.

- Говоришь, не похоже... Ну-ка, смотри: что это?

- Это? Самолет, разумеется. — Да. А это? — Тоже самолет, только другой: истребитель.

- Ну если одно — самолет, то второе — не самолет: они же не похожи.

- Ну да, не похожи, скажешь тоже!

- Смотри сам. Нос похож? Нет. Крылья похожи? Нет! Двигатели похожи? Тоже нет. А что похоже?

- Хм... Действительно. Вроде все по отдельности не похоже, а вместе похоже, и сразу видно, что и то и другое — самолет.

- Ну вот так же и с ЭВМ — все по отдельности не похоже, а вместе и то и то — ЭВМ. Ну-ка, давай прочтем, что там написано в учебнике. Так, во-первых ЭВМ состоит из процессора, выполняющего программу, памяти, хранящей программу и данные, и устройств ввода и вывода. Процессор на ПМК есть?

- Есть, по-видимому, хотя его снаружи и не видно: что же в нем выполняет программу.

- Хорошо. Память есть?

- Есть и даже несколько видов:

регистры стека, адресуемые регистры и программная память.

- Это уже детали. Главное, что есть. Устройства ввода и вывода есть?

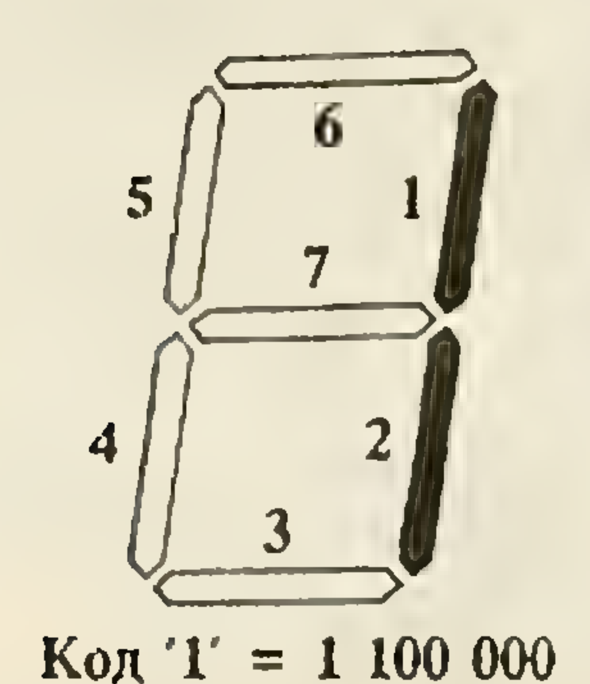
- Ну разве что клавиатура и индикатор. Тоже мне устройства! Индикатор же только цифры и выводит.

- Это уже второй вопрос. А чем тебе индикатор плох — нигде же не написано, что именно это устройство должно выводить. Итак, как видишь, по определению ЭВМ калькулятор вполне проходит как ЭВМ. Ну а теперь давай смотреть детали. Что там первое не похоже?

- Во-первых, данные представляются в какой-то двоичной системе и состоят сплошь из нулей и единиц. Я не совсем понял, что это такое и почему именно двоичная система. А на ПМК все пишется как обычно.

- Ну почему двоичная система, объяснить легко: очень просто кодировать. Там ведь всего два символа: ток есть — это единица, тока нет — ноль. Или намагниченность в ту или другую сторону — это ноль или один. А на ПМК все, кстати, тоже кодируется в той же двоичной системе. Смотри, вот у тебя символ на индикаторе — он состоит из семи светящихся сегментов. Занумеруем сегменты как угодно. Теперь для того, чтобы "объяснить" индикатору, какую цифру надо зажечь, ему надо передать, какие сегменты должны засветиться. Чтобы высветить цифру ´1´ надо зажечь сегменты 1 и 2 и потушить остальные. Получаем код цифры ´1´ — 1100000, т.е. единички на местах номер 1 и 2 показывают, что сегменты 1 и 2 должны гореть, а на остальных местах нолики, т.е. сегменты должны быть потушены. Точно так же можно выписать коды и для остальных цифр. Причем, обрати внимание, если по-разному нумеровать сегменты, то получатся разные коды. Понял?

- Вроде да. А в учебнике код из восьми цифр. Ну это уже мелочи.



Код ´1´ = 1 100 000

Но числа кодируются совсем не так.

- Там числа кодируются так, чтобы их удобнее было складывать, умножать и т.д. А при выводе на индикатор как не нумеруй сегменты рисунка, цифры не получишь. Но большим ЭВМ это и не надо. Кстати, и на ПМК номер регистра А выглядит на индикаторе как минус, а не как буква "А". А вообще про двоичную систему, равно как и про другие, можешь почитать книжку Фомина "Системы счисления". Что дальше в учебнике написано?

- Каждый символ требует для хранения 1 байт — это такая единица памяти из 8 битов. А каждые 2 байта образуют слово, которое может хранить команду.

- Ну здесь вообще все один к одному, как на ПМК. Цифра — это 1 байт, только из 7 битов, но это тебя, надеюсь, не смущает. Ячейка программной памяти — это слово, те же 2 байта, и может хранить команду.

- Смотри-ка, действительно, все очень похоже. Так, что там дальше? Регистры — есть! Только на ЭВМ их 8, а на ПМК — 15.

Счетчик адреса команд на ПМК — это программный счетчик на ЭВМ.

- Только на ПМК это совсем особый регистр, а на ЭВМ его теоретически можно использовать как обычный регистр, хотя никто этого не делает. Надеюсь, тебя не смущает, что на ЭВМ регистры и ячейки памяти одной длины, а на ПМК ячейки программной памяти короткие, а регистры длинные?

- Это меня не смущает, а вот команды на ЭВМ ну какие-то совсем не похожие на команды ПМК.

- Они не похожи не более, чем крылья биплана на крылья истребителя. Просто на ДВК-2 двухадресные команды, т.е. в каждой команде задано два места: откуда брать данные и куда класть результат, а на ПМК команды либо одноадресные, либо вообще безадресные, т.е. задано только одно место, а второе ПМК знает сам: это всегда стек. Куда пересылает число команда "П-x"? В стек. А откуда брать число — это ей указано. А команда " + " вообще адресов не имеет: данные всегда в стеке и результат там же. Давай сравним программы для ДВК-2 и для ПМК.

Программа для ЭВМ ДВК-2		Программа для ПМК
Команда	Адрес	Команда
переслать слово R0 в R2	1500	П-х 0 х-П 2
добавить слово R1 к R2	1502	П-х 2 П-х 1 + х-П 2
стоп	1504	С/П

- Ну как, похоже?

- Вообще похоже. Но только команды какие-то странные — пока напишешь — устанешь: "переслать слово ...", на ПМК все же как-то проще.

- Ну это в учебнике их так пишут, а в настоящем программировании есть сокращенная запись, которой, собственно программисты и пользуются. Никто не пишет так полными словами. Пишут, например,

MOV R0,R2

Слово "MOV" — это сокращение от английского "move" — "переслать". Так как, похоже?

- Ну так совсем похоже. Ладно, тогда пошли дальше. Там для сравнения чисел какие-то биты N и Z используются. Это что такое?

- Бит N — это просто знак "минус" на индикаторе. Когда результат отрицателен, то этот знак горит, т.е. соответствующий этому знаку бит равен единице. А вот бит Z на ПМК не виден, хотя он тоже есть. Команда ПМК "F X=0" как раз и проверяет этот бит и выполняет переход, если он потушен, т.е. нулевой. А команды "F X>=0" и "F X<0" проверяют бит N. Ну что, сравним программы для ДВК и ПМК? Вот программа, определяющая минимальное из чисел, хранящихся в R1 и R2, и циклическая программа, вычисляющая

$$K + (K-1) + (K-2) + \ldots + 1.$$

(Обе программы взяты из школьного учебника.)

- Да, похоже и весьма. Только команды перехода какие-то плохие: неудобно считать, на сколько слов переход надо сделать. Зачем такую хитрость сделали?

- А ты видишь, что в программе

Программа для ЭВМ ДВК-2		Программа для ПМК
Команда	Адрес	Команда
сравнить слово R1 с R2	1500	П-х 2 П-х 1 −
если меньше, переход на +2 слова	1502	Fx>=0 +——+++++
переслать слово R1 в R3	1504	П-х 1 х-П 3
переход на +1 слово	1506	БП ++——+++++
переслать слово R2 в R3	1508	+-> П-х 2 х-П 3
стоп	1510	+-> С/П
очистить регистр R2		Cx х-П 2
переслать слово R1 в R0		П-х 1 х-П 0
добавить слово R0 к R2		+-> П-х 2 П-х 0 + х-П 2
цикл по регистру R0, на -2 слова		F L0 +——++++
стоп		С/П

для ПМК адреса мы написали условно? Почему?

- Но они ведь зависят от того, с какого адреса написана программа.

- Верно. А что изменится в программе для ДВК, если ее написать с другого адреса?

- А-а, понял! В программе для ДВК ничего не меняется, с какого адреса ни помести этот кусок.

- Точно. Для больших ЭВМ это существенно. Такая запись называется программированием в относительных адресах, потому что адреса считаются не от начала всех ячеек памяти, а относительно адреса текущей команды.

- Ладно, сдаюсь. Действительно получается, что калькулятор — это ЭВМ. Так, значит, освоив программирование на ПМК, можно очень легко освоить программирование на любой ЭВМ?

- И ты только сейчас это понял? Ладно, хорошо что понял вообще. Конечно, ПМК не идет ни в какое сравнение с большими ЭВМ, но принципы в них заложены те же. И тот, кто освоил ПМК, безусловно, сделал огромный шаг к компьютерной грамотности.


EPSON



Они везде извека:
В планетах, днях, природе,
В твореньях Человека,
В лесу и в огороде.
И даже в шишках ели,
И в стеблях ананаса
(Вы их, конечно, ели)
Этих чисел масса.
Без них не ждать удачи,
Я за, пока не сгину,
За числа Фибоначчи,
За златосередину!

Б.А.Тарасенко

ТРИЖДЫ ФИБОНАЧЧИ!

Выпуск 3

Человечество не удивишь идолами, математической мистикой, слепой верой. Но если когда-нибудь возникнет вновь культ чисел Фибоначчи, то, по нашему глубокому убеждению, это будет самая разумная, самая естественная и человечная религия в мире. Потому что жизнь — это оптимизация и наоборот, а числа Фибоначчи — это одно из проявлений великого закона оптимизации, закона жизни, развития, прогресса. В новой вере к лику святых безусловно будут отнесены Пифагор, Леонардо Пизанский сын Боначчи, фра Лука Пачоли, Леонардо да Винчи, Альбрехт Дюрер. Наука даст новых добрых пастырей. Прихожанами станет большинство населения Земли.

О числах Фибоначчи, казалось бы, сказано почти все в несколько раз переизданной работе [1]. Но смелым умам нет предела. Например, автор книги [2] Стахов А.П. нашел, что можно добавить к международному математическому журналу, звучащему в переводе с английского примерно как "Ежеквартальник Фибоначчи". Оказалось, что при прочих равных условиях, супер-ЭВМ будет функционировать "чуть" быстрее, "чуть" точнее, надежнее, если числа и команды будущей машины закодировать в системе, основанной на этом удивительном математическом феномене [3]. Если же "рацпредложение" будет еще и внедрено, то усилиями общественности, в том числе газеты "Правда", нашей стране прибавится "чуть" больше международного научного престижа. И не только научного.

Принцип оптимизации обнаруживается прежде всего в объектах живой природы, а вообще повсюду, на всех уровнях, которые кажутся и значительными [2, 3], и незначительными [4, 5]. Мы не удивимся, если "божественная" пропорция проявится на частотах, соответствующих гармоничной живописной цветовой комбинации, в распределении по стоимости и качеству однотипных продуктов в здоровом обществе и так далее. Даже в букете приятных запахов.

Программы и алгоритмы не могут существовать без своих носителей, качество и ценность программ невозможно понять без устройств, на которых они реализуются, т.е. без исполнителей. Поэтому невозможно говорить о правильной цене на программу, если она оторвана от цены носителя и, что особенно важно, от цены исполнительного устройства. Человек — всегда соисполнитель ЭВМ, причем ведущий, и стоимость примененной программы зависит от его компетентности.

Если массовые учебные программы красиво вписываются в возможности массового же калькулятора МК-54 стоимостью 65 рублей, то естественно ли, что набор подобных учебных программ кое-кто пытается воспроизводить на настольных "персоналках" стоимостью в 50 тысяч рублей? Нет, конечно. Большому кораблю — большое плавание, малой флотилии — другая, но не менее почетная и важная служба. Естественно ли, что Всесоюный фонд алгоритмов и программ не признает за программный продукт (с защитой авторских прав) математическое обеспечение к МК-54, МК-61, в то время как, например, фирма "Хьюлетт-Паккард" имеет многочисленные пакеты прикладных программ к подобным карманным компьютерам и реальную прибыль от их продажи? Нет, ситуация абсурдна.

В сложившемся положении напрашивается вывод, о том, что хорошо систематизированные серийные публикации по программному обеспечению калькуляторов в какой-то степени защищают авторские права создателей программ и берут на себя функции фондов программ. И мы с вами, уважаемые читатели, на верном пути.

Казалось бы, автора опять "занесло" в сторону от темы. На самом деле мы с вами вместе наблюдаем естественный процесс (и участвуем в нем!) социальной оптимизации по проблеме всеобщей компьютерной грамотности. А оптимизация — это тот же принцип, те же числа, о которых мы сейчас ведем разговор.

Но вернемся к нашей основной теме и трижды воздадим хвалу первому, кто упомянул о "кроличьих" числах [1], великому Леонардо из Пизы, сыну Боначчи.

Программа № 8

Демонстрационный генератор чисел Фибоначчи
с полной автоподготовкой

```
АА   АЛГ ГЕНЧИСФИБ1 /М, Б/    / + заголовок, переменные              + /
ББ        АРГ М, Б            / + аргументы при вводе                + /
ВВ        РЕЗ Б,М             / + результаты при выводе              + /
ГГ ПОЯСНЕН В алгортме ГЕНЧИСФИБ1, или генераторе чисел Фибона-
   ччи № 1, используются два соседних числа Фибоначчи: М — мень-
   шее, Б — большее, а в каждом цикле вычислений выполняется Р: = Б,
   Б: = Б + М, а затем М: = Р, Р — рабочая переменная и ячейка; инди-
   цируется только число Б. М = сPrX1,Б = сPrX. Пригодны: МК-61, МК-52,
   Б3-34, МК-54. Начальные аргументы вводятся автоматически из
   текста программы.
ДД   ВСЕ о пояснениях / + конец описаний и пояснений             + /
ЕЕ   НАЧ              / + начало собственно алгоритма            + /
ЖЖ   ВКЛ-ВКЛ          / + включ. питания, высвеч. 0. слева       + /
ЗА   Ф ПРГ   / + переход в реж. записи про-мм, высв. 00 справа   + /
ЗБ   ВЫПОЛНИТЬ / + запись следующего ниже текста программы       + /
  00 1         01 / + КЛАВКО 1, или клавишная константа 1        + /
  01 Ф 1/Х     23 / + после 1/Х сPrX остается равным единице     + /
  02 0         00 / + КЛАВКО 0; сPrX = 0, сPrУ = 1, сPrX1 = 1    + /
  03 Ф Вх       0 / + сPrX1 на PrX; сPrX = М                     + /
  04 + - +     14 / + встречный обмен сPrX и сPrУ; сPнХ = Б      + /
  05 +         10 / + новое Б: = старое Б + старое М, новое М = сPrX1 + /
  06 С/П       50 / + стоп-индикация Б, большего из двух чисел   + /
  07 БП        51 / + безусловный переход, заголовок команды     + /
  08 0 3       03 / + адрес безусловного перехода                + /
ЗВ   ПОЯСНЕН Конец ручного набора текста программы
ЗГ   ВСЕ о пояснениях
ИИ   Ф АВТ       / + переход в реж.исполнения, 0.справа          + /
КК   В/О / + установка счетчика шагов на нулевой начальн.адр.    + /
ЛЛ   С/П / + пуск программы на счет                              + /
ММ НАБЛЮДЕНИЕ / + мигание около одной секунды, затем            + /
              / + ровное высвечивание индикатора; в это          + /
              / + время можно визуально считать очеред-          + /
              / + ное число Фибоначчи                            + /
НА   ЕСЛИ нужно продолжить счет
НБ   ТО ХОД на метку ЛЛ
НВ   ИНАЧЕ ВКЛ-ОТКЛ / + отключение питания, гасн.инд.            + /
НГ   ВСЕ
НД   КОН алгоритма
```

В программах №№ 8 и 9 применены уже знакомые читателям по вып.2 особые комментирующие скобки. Для наглядности и для напоминания о способе их использования приводится пример / + ВОТ ВАМ И ВНУТРИСКОБОЧНЫЙ "ПРИМЕРНЫЙ" ТЕКСТ + /. Комментирующая запись внутри подобных скобок, как правило, относится к соответствующей командной строке, расположенной левее скобок. Опустим в записи (но не в исполнении!) ручные команды Ф ПРГ Ф АВТ В/О, место и роль которых, полагаем, читатели твердо усвоили.

Для записи инструкций к программам здесь применен школьный безмашинный алгоритмический язык /ШБАЯ/, в который введены некоторые добавления и которые, надеемся, будут понятны без объяснений. Дело в том, что инструкция — это тоже алгоритм, где художественная проза подходит меньше всего. Так почему бы для описания инструкции, или последовательности ручных команд, не использовать детерминированную краткость ШБАЯ? Именно это мы и сделали. Нам кажется, что ШБАЯ по отношению к калькулятору превращается, таким образом, из красивого, но неприкаянного "летучего голландца" в практичного и понятного трудягу, обретающего наконец твердую почву под ногами. Переписывать же программы в клавишном языке калькулятора из ШБАЯ и наоборот — это нечто противоестественное.

В программах 8 и 9 клавишный язык калькулятора выглядит как автокод по отношению к языку высокого уровня, а именно по отношению к языку ШБАЯ, а все это сочетание в целом

Программа № 9

Счетный генератор чисел Фибоначчи

```
АА АЛГ ГЕНЧИСФИБ2 /ЦЕЛ М, Б, С/ / + заголовок алгоритма          + /
ББ  АРГ М, Б, С            / + входные целые переменные, из програ-ы  + /
ВВ  РЕЗ Б, М, С            / + результирующие переменные при выводе  + /
ГА  ПОЯСНЕН М — меньшее, Б — большее из двух соседних чисел
       Фибоначчи, С — порядковый номер числа Б.
ГБ  ВСЕ о пояснениях'
ДД  НАЧ                    / + начало собственно алгоритма             + /
  00  1              01    / + КЛАВКО 1, или клавишая константа 1      + /
  01  Х- + П А       4-    / + М: = 1, так как сРгА = М, так принято    + /
  02  0              00    / + КЛАВКО 0, или клавишн.константа 0       + /
  03  Х- + П В       4Л    / + Б: = 0, так как принимаем сРгВ = Б       + /
  04  Х- + П 4       44    / + С: = 0, или сРг4 = 0, или нач.уст.счет.  + /
  05  К П- + Х 4     Г4    / + С: = С + 1, или работа счетчика         + /
  06  П- + Х А       6-    / + сРгХ: = М с вызовом из РгА              + /
  07  П- + Х В       6Л    / + сРгХ: = Б с вызовом из РгВ, сРгУ: = М    + /
  08  Х- + П А       4-    / + сРгА: = новое М, новое М = старое Б      + /
  09   +             10    / + Б: = М + Б, справа от знака " = " стар.зн. + /
  10  Х- + П В       4Л    / + сРгВ5: = новое Б, сРгХ = Б               + /
  11  П- + Х 4       64    / + сРгХ5: = сРг4, или счетчика, сРгУ: = Б   + /
  12   + - +         14    / + встречный обмен сРгХ и сРгУ; сРгХ = Б    + /
  13  С/П            50    / + стоп-индикация Б; возмсжн.вызова С       + /
  14  БП             51    / + БП, или заголовок команды безусл.перех.  + /
  15  0 5            05    / + переход по БП на шаг 05                  + /
ЖЖ  С/П                    / + ручной пуск программы на автомат.счет    + /
ЗА  НАБЛЮДЕНИЕ             / + мигания около 5 секунд                   + /
ЗБ  СТОП-ИНДИКАЦИЯ / + очередного числа Фибоначчи с ост.счета           + /
ИИ        ЕСЛИ нужен порядковый номер числа Фибоначчи,
КК              ТО + - + / + через секунду высвечивается этот номер     + /
ЛЛ        ВСЕ              / + что касается номера числа Фибоначчи      + /
ММ        ЕСЛИ нужно продолжить счет
НН              ТО ХОД на метку ЖЖ
ОА              ИНАЧЕ ВКЛ-ОТКЛ  / + отключение питания                  + /
ОБ        ВСЕ
ОВ  КОНЕЦ алгоритма ГЕНЧИСФИБ2
```

смотрится, на наш взгляд, вполне гармонично. Естественно применить здесь и принцип умолчания, что и сделано для программы № 9 по отношению к командам перехода из режима в режим и установки пускового адреса программы. Так, если адрес пуска не указан, то это означает, что он нулевой и что для установки нулевого начального адреса нужно перед пуском программы на счет выполнить ручную команду В/О. Способ маркирования команд, ручных и записанных в программной памяти калькулятора, напоминает механизм маркирования строк в БЕЙСИКе. Все показанное — это демонстрация возможностей, а не призыв к массовому повторению, поэтому инструктивная часть к программе № 10 дается одним из старых способов.

Все три программы показывают числа Фибоначчи по абсолютной величине от 1 до 39 — точно, а начиная с 40 и выше — приближенно. По структурной организации все они являются "бесконечными" демонстраторами чисел. Программы снабжены внутренней самоустановкой и перед началом работы требуют только однократной ручной команды В/О. Далее на каждое нажатие клавиши С/П будут появляться уже знакомые вам числа 1, 1, 2, 3, 5, 8, 13, 21, ... Для программ 9 и 10 при останове счета номер индицируемого числа может быть вызван на индикатор ручной клавишной командой П- + Х 4. Очередное нажатие С/П — и счет может быть продолжен вслед за этим без всякой дополнительной начальной установки.

Если желательно сменить направление генерации фибоначчиева ряда, то это возможно только для программы 10. Для осуществления реверса вслед за наблюдением очередного числа Фибоначчи и перед очередным нажатием С/П необходимо один раз нажать на клавишу ШГВ (или "шаг вперед", или ШГ). Нажатие можно сделать в любом "месте" числового ряда и менять направление генерации многократно. Если порядковые номера дать в верхней строчке, а числа Фибоначчи — в нижней, то для программы 10 можно получить примерно следующую диаграмму результатов ее работы:

```
... -7   -6   -5   -4   -3   -2   -1   0  + 1  + 2  + 3  + 4  + 5  + 6   + 7 ...
... -13  8    -5   3    -2   1    -1   0    1    1    2    3    5    8   13 ...
```

Программа № 10

Счетный реверсивный генератор чисел Фибоначчи

00	1		01	/ + КЛАВКО 1, или клавишная константа, начало + /
01	1		01	/ + КЛАВКО 11, конец "автонабора" + /
02	Х- + П		4С	/ + сРгС: = 11, или засыл числа 11 в РгС + /
03	2	С	02	/ + КЛАВКО 2, начало "автонабора" из программы + /
04	9		09	/ + КЛАВКО 29, завершение "автонабора" + /
05	Х- + П	Д	4Г	/ + сРгД: = 29, или засыл числа 29 в РгД + /
06	Сх		ОГ	/ + сРгХ: = 0 + /
07	Х- + П	4	44	/ + сРг4: = 0, или С: = 0, или обнуление счетчика + /
08	Х- + П	В	4Л	/ + сРгВ: = 0, или Б: = 0; Б — большое число + /
09	1		01	/ + КЛАВКО 1 + /
10	Х- + П	А	4-	/ + сРгА: = 1, или М: = 1;М — меньшее число + /
11	П- + Х	4	64	/ + сРгХ: = сРг4, или вызов на РгХ содерж.Рг4 + /
12	1		01	/ + КЛАВКО 1 + /
13	+		10	/ + сРгХ: = сРгХ + 1, фрагмент работы счетчика + /
14	Х- + П	4	44	/ + сРг4: = сРгХ, или С: = С + 1, счетчик пор. №№ + /
15	П- + Х	А	6-	/ + вызов на РгХ меньшего текущего числа ШМ + /
16	П- + Х	В	6Л	/ + вызов на РгХ большего текущего числа Б + /
17	Х- + П	А	4-	/ + новое М = старому Б, фиксация в РгА + /
18	+		10	/ + новое Б = старое Б + старое М + /
19	Х- + П	В	6Л	/ + сРгВ: = новое Б путем засылки в РгВ + /
20	С/П		50	/ + стоп-индикация Б, или большего числа + /
21	К БП	С	8С	/ + ХОД на шаг, или адрес, по содерж. РгС + /
				/ + здесь, точнее при стоп-индикации, возможен реверс за счет ручной ШГВ-клавиши + /
22	П- + Х	С	6С	/ + сРгХ: = сРгС + /
23	П- + Х	Д	6Г	/ + сРгХ: = сРгД; сРгУ: = сРгС + /
24	Х- + П	С	4С	/ + сРгС: = сРгД, первая фаза обмена + /
25	+ - +		14	/ + встречный обмен сРгХ и сРгУ + /
26	Х- + П	Д	4Г	/ + сРгД: = сРгС, вторая фаза обмена между РгС и РгД, смена базы косвенной адрес-и + /
27	БП		51	/ + БП, или заголовок команды безусловн.пер, + /
28	2 1		21	/ + безусловный переход на шаг 21 + /
29	П- + Х	4	64	/ + вызов сРг4 на РгХ, или счет. порядк. №№ + /
30	1		01	/ + КЛАВКО 1 + /
31	—		11	/ + С: = С-1, или перв. фаза раб. сч.на вычит. + /
32	Х- + П	4	44	/ + засылка нов. знач. С в Рг4, регистр-сч. + /
33	П- + Х	В	6Л	/ + вызов старого Б на РгХ + /
34	П- + Х	А	6-	/ + вызов старого М на РгХ, засыл Б в РгУ + /
35	—		11	/ + образование разности Б — М, или нового М + /
36	Х- + П	А	4-	/ + фиксация нового М в РгА + /
37	Ф Вх		0	/ + вызов старого М, или нового Б + /
38	Х- + П В		4Л	/ + фиксация нового Б в РгВ + /
39	БП		51	/ + БП, заголовок команды безусловного пер. + /
40	2 0		20	/ + безусловный переход на шаг 20 + /

Нам кажется, что представленный набор программ будет полезен, особенно последняя, не только для учебы, но и для стимуляции новых результативных поисков по затронутой теме, математических и практических. При переходе через точную границу и последующем возврате к нулю (программа 10) будет "захвачена зарубежная неточность". "Неточность возврата", видимо, будет зависеть от глубины перехода плюсовой или минусовой области нумерации и, предположительно, от предыстории состояния некоторых регистров калькулятора. Исследователи, вам в руки фигуральные "карты" и реальная программа!

Чтобы легче было вести такие исследования, приводим таблицу разложения чисел Фибоначчи на простые сомножители. Она составлена честным путем, т.е. с использованием калькулятора не мощнее МК-85 и с помощью собственноручно составленных вспомогательных программ. В популярной литературе до столь "далеких" порядковых номеров еще, кажется, не доходили, ну, мы и решили заполнить этот пробел... Решили заодно напомнить об одном интересном свойстве чисел Фибоначчи, которое лучше наблюдать как раз на таблице увеличенного объема. Суть свойства: если больший порядковый номер "кроличьего" числа кратен меньшему, то соответствующее большее число Фибоначчи кратно меньшему. Например, если

14 и 7 есть порядковые номера, то соответствующие им фибоначчиевы числа 377 = 13 x 29 и 13 кратны 13. Остальные примеры, уважаемые читатели, вы во множестве можете подобрать сами. При составлении приведенной таблицы упомянутым свойством делимости мы очень широко пользовались.

Разложение чисел Фибоначчи на простые сомножители

Порядковый номер числа Фибоначчи	Число Фибоначчи		Разложение на множители
1	1	=	1
2	1	=	1
3	2	=	2
4	3	=	3
5	5	=	5
6	8	=	2x2x2
7	13	=	13
8	21	=	3x7
9	34	=	2x17
10	55	=	5x11
11	89	=	89
12	144	=	2x2x2x2x3x3
13	233	=	233
14	377	=	13x29
15	610	=	2x5x61
16	987	=	3x7x47
17	1597	=	1597
18	2584	=	2x2x2x17x19
19	4181	=	37x113
20	6765	=	3x5x11x41
21	10946	=	2x13x421
22	17711	=	89x199
23	28657	=	28657
24	46368	=	2x2x2x2x2x3x3x7x23
25	75025	=	5x5x3001
26	121393	=	233x521
27	196418	=	2x17x53x109
28	317811	=	3x13x29x281
29	514229	=	514229
30	832040	=	2x2x2x5x11x31x61
31	1346269	=	557x2417
32	2178309	=	3x7x47x2207
33	3524578	=	2x89x19801
34	5702887	=	1597x3571
35	9227465	=	5x13x141961
36	14930352	=	2x8x3x3x3x17x19x107
37	24157817	=	73x149x2221
38	39088169	=	37x113x9349
39	63245986	=	2x233x135721
40	102334156	=	3x5x7x11x41x2161
41	165580141	=	2789x59369
42	267914296	=	2x2x2x13x29x211x421
43	433494437	=	433494437
44	701408733	=	3x43x89x199x307
45	1134903170	=	2x5x17x61x109441
46	1836311903	=	139x461x28657

Демонстрационные программные генераторы чисел Фибоначчи, приведенные в [4], отличаются от аналогичных программ 8, 9 и 10 тем, что у них предъявляемый отрезок фибоначчиева ряда укорочен, т.е. отсутствует одно или даже два первых числа. Интервал времени между нажатием С/П и появлением очередного числа на экране индикатора для новых программ, с учетом разброса быстродействия разных калькуляторов, может оказаться равным соответственно 2, 3 и 5 секундам.

Существует еще одно золотое правило, явно находящееся в родстве с божественной пропорцией: после относительно сложной работы необходимо переходить к менее сложной. Чтение наших статей рубрики нельзя считать развлекательным, поэтому последуем за природой, а не вопреки ей.

Чертова дюжина

Задумаемся об одном примечательном совпадении. Случайно ли, что один из первых "толкователей" принципа оптимизации, Фибоначчи, жил в XIII веке? Что легендарный автор основных канонов социального переустройства общества, Иисус Христос, в кульминационный момент своей жизни подобрал себе группу сподвижников (вместе с ним самим) в числе 13? Случайно ли, что круговой манеж практически всех цирков мира почти в точности равен 13 метрам? Что жрецы беззастенчиво меняли продолжительность марцедония — 13 месяца древних римлян? Что в календарях майя было 13 месяцев? Что, наконец, число 13 есть среди чисел Фибоначчи и на многих живых объектах растительного происхождения? Предполагаем, что так получается совсем не случайно и что в мире животных число 13 тоже существует на вполне законных основаниях. Может статься, что это редкая морская звезда с 13 лучами, фораминифера, или млекопитающее с 13 парами зубов. Не исключено, что числом 13 отмечен организм, который уже был на земле, но которого уже нет в текущей жизни, возможно, что ему еще только предстоит появиться в цепочке биоэволюции. Много ли мы знаем в подобном плане о насекомых, тем более — о микроорганизмах? Вероятно, что некоторое ведущее число в строении тела животного окажется равным или кратным 13, например число пар ног у некоторых видов "тысяченожек". Есть же иные совпадения: у паукообразных по восемь ног — число Фибоначчи! У насекомых по шесть ног, 6 = 2x3 — произведение двух соседних чисел Фибоначчи! У человека по пять пальцев на каждой конечности, наиболее известны пятилучевые морские звезды, цветы семейства розоцветных имеют по пять лепестков. Видимо, не случайно такой феномен увековечен природой, так как пять — это единственное число Фибоначчи, у которого его порядковый номер совпадает с ним самим.

Чем меньше животное, тем больше количество приплода за один помет. А каково типовое число детенышей за один помет у разных животных, яичек — в кладках птиц, бабочек, жуков, стрекоз? Каково количество сосков на молочной железе у разных видов млекопитающих, как это число связано с типовым количеством разового приплода? У утконоса и ехидны сосков нет вообще, молоко свободно изливается на покрытую шерстью часть живота, откуда детеныш и слизывает ценную материнскую пищу. Чем не две единицы в начале ряда Фибоначчи! В подобном аспекте живая модель фибоначчиева числа два известна всем и обсуждению не подлежит. Трудности наступают с числами "три" и "пять". И все же, вдруг в природе существует редкий вид, например, опоссума, самки которого имеют по 13 сосков?

Напишите нам о научно подтверждаемых фактах такого рода, спросите у знакомых биологов, потому что они редко бывают одновременно программистами и математиками.

Оставаясь в рамках, очерченных основным заголовком выпуска 3, рассмотрим проблему еще с одной стороны.

Знают ли о числах Фибоначчи другие?

К сожалению, об этом знают очень немногие. Поэтому любую объективную пропаганду, публикации на эту тему следует только приветствовать. Как и подборку программ о числах Фибоначчи в журнале "Наука и жизнь" № 4 за 1990 год, стр. 97. Правда, с оговорками. Числа Леонардо наконец-то возвращаются в школьную программу. Программа, которая вычисляет число Фибоначчи по заданному его номеру, совпадает по назначению и организации с нашей программой № 008 [4]. Программа хороша, компактна. Однако нам удалось найти небольшую "щель" в этом монолите, сократив программный текст на две команды и сделав ее более удобной в использовании. Приводим эту программу в нашем варианте.

Программа № 11

Генератор числа Фибоначчи по его заданному номеру

00	Х-+П 0	40	06	+	10
01	1	01	07	Ф ЛО	5Г
02	+-+	14	08	0 4	04
03	0	00	09	С/П	50
04	Ф Вх	0	10	БП	51
05	+-+	14	11	0 0	00

Инструкция:

В/О, набор порядкового номера Н искомого числа Фибоначчи, С/П; набор нового номера, С/П и так далее.

Программа работает точно в диапазоне порядковых номеров от 1 до 39, для номеров выше 39-го числа выдаются с округлением. На поиск последнего точно вычисляемого числа, 63245986, на МК-52 было затрачено 53 с. Нельзя сказать, что очень уж большое время. Программа получилась короткой и быстрой.

В темплане издательства "Наука" по физико-математической литературе на 1990 год под № 25 значится шестое издание книги Воробьева Николая Николаевича "Числа Фибоначчи". В книге имеется "... приложение, содержащее программы на языке БЕЙСИК для всех приведенных в книге алгоритмов". Издательство "Наука" в рекомендациях не нуждается, и эту научно-популярную книжку читателям стоит взять на заметку. По части приложения и БЕЙСИКа для МК-85 мы можем оказаться даже несколько впереди.

Программа № 12

Простейший демонстрационный счетный генератор чисел Фибоначчи

```
1 PRINT "ГЕН.ФИБОНАЧЧ"
2 A = 1 : B = 0 : N = 0
3 N = N + 1
4 C = A:A = B:B = C + A
5 PRINT "N"; N, "F = "; B
6 GOTO 3
```

После того как вы запишите текст программы 12 в один из десяти файлов МК-85, например в файл Р2 (на клавишной панели — над клавишей, красным цветом) и перейдете в режим автоматического исполнения (MODE O), вам достаточно нажать клавишу "S латинское красное", затем клавишу 2 (Р2) и на экране индикатора появится надпись ГЕН.ФИБОНАЧЧ. Далее остается только нажимать клавишу EXE. При каждом нажатии на экране будет появляться сначала номер, помеченный соответствующей буквой, а после него — тоже помеченное буквой сопряженное число Фибоначчи. Программа работает точно до Н = 49 $Фу_9$ = 7778742049 включительно. Начиная с порядкового номера 50, числа Фибоначчи вычисляются приближенно и высвечиваются в степенной форме записи. После нажатия EXE результаты на экране появляются практически мгновенно. При скоростном режиме (а для МК-85 возможен и такой) вычисления производятся в 6 раз быстрее обычного всего лишь при 4-кратном увеличении потребления по току (примерно с 2 до 8 мА). Лишь начиная с 45-го числа вам придется ждать 1 — 2 секунды, до тех пор пока слишком большое число на экране автоматически не продвинется, после паузы, справо напево (буквы при этом выдвигаются "за пределы" экрана).

В "Справочнике по расчетам на микрокалькуляторах" В.П.Дьяконова (3-е изд. — М.: Наука, 1989) не без труда разыскиваем на стр. 180 программу 4.28 "Поиск числа Фибоначчи по заданному номеру". Первый сюрприз: если мы вводим номер, то число, выдаваемое программой, оказывается связанным с номером, который на единицу больше исходного. Не очень удобно, но нас предупредили. Вводим и проверяем программу. Здесь нас ждет сюрприз посильнее. Оказывается, что по справочнику одиннадцатое число равно 144, когда оно на самом деле меньше и равно 89. Но неправильно напечатано, работающая программа подтверждает! Все сдвинуто на единицу! Так нельзя, это закон математики. Напечатанное равносильно объявлению: "С завтрашнего дня считать 2х2 = 5."

Обнаружив опечатки и небрежности в других программах, припомнив предупреждения знакомых преподавателей вузов, приходим к выводу, что здесь, в справочнике, еще о числах Фибоначчи не знают. У справочника третьего издания большой "плюс": широкий охват тем. Но читателю придется соблюдать девиз: не проверив, не пользуйся. А что-то придется сделать самостоятельно.

Возьмем заводское "Руководство по эксплуатации" к микрокомпьютеру "Электроника МК-85". На страницах 49 и 50 обнаруживаем... программу нахождения члена ряда Фибоначчи по заданному номеру. Вводим программу, запускаем при порядковых номерах 0, 1, 2 и 3 и каждый раз получаем единицы. Что-то многовато единиц для одного ряда Фибоначчи. Возможно, что мы сделали не очень корректный ввод. Да, согласны. Но для неподготовленного пользователя в инструкции никаких предупреждений нет. Идем далее, т.е. набираем порядковый номер 4, получаем число 2. Очень интересно! Что дальше? Для порядкового номера 11 получаем уже не 89, как должно быть, не 144, как у Дьяконова В.П., а 55. Похоже, что здесь, неправильно понимая "компьютерный плюрализм", "играют на понижение" и готовы, не моргнув глазом, объявить, что 2х2 = 3.

Вежливые и воспитанные читатели, а также очень предусмотрительные, посетуют в том смысле, что не стоило бы так обижать наших "благодетелей", которые сделали неплохой по устройству и цене микрокомпьютер. Но где они, эти микрорадости? Почему ими не завалены полки наших магазинов, школы и детские сады, НИИ и вузы? Почему столь малы годовые планы выпуска МК-85 и куда идет сверхплановая продукция? Вопросы не риторические и возникающие не от избытка полемического задора. Действительно, МК-85 может быть полезен и ребенку, и академику, школьнику, студенту, инженеру. Все эти вопросы (и многие другие) от редакции и от имени наших читателей (в редакции есть много писем с читательскими вопросами по поводу МК-85) мы хотели задать представителю завода — изготовителя этого дешевого карманного компьютерного чуда. Но представителю редакции было отказано в чисто техническом интервью. У нас еще будет время написать, опубликовать много красивых и полезных программ, а у представителей разработчиков МК-85 — подумать и, возможно, изменить свою позицию. Читатели будут ждать и того, и другого. Мы уверены.

Заканчивая этот выпуск, сообщим, что у нас много читательских писем с предложениями, вопросами, программами. Наиболее интересное из этой почты найдет место на страницах последующих выпусков. Но следует предупредить, что в первую очередь рассматриваются предложения, напечатанные на машинке (или принтере) с аккуратно написанными (лучше напечатанными) программами.

Литература

1. Воробьев Н.Н. Числа Фибоначчи. — М.: Наука, 1978. — 144 с.

2. Стахов А.П. Коды золотой пропорции. — М.: Радио и связь, 1984. — 152 с.

3. Реут В. Вот вам и Фибоначчи!//Правда. — 1988. — 21 ноября.

4. Тарасенко Б.А. Алгоритмический букварь и карманная ЭВМ. — Вып. 4. — Серия "Вычислительная техника и ее применение". — М.: "Знание", 1988.

5. Тарасенко Б.А. Делитель частоты. Авторское свидетельство № 343386. БИ № 20 от 33 июня 1973 года.

Программа — Предельно короткий нарастающий генератор простых чисел

Автор: Тарасенко Б.А. Программа разработана 5.08.89. Публикуется впервые. Реализуема на калькуляторах МК-61 и МК-52. Назначение: учебная программа для кружков программирования и математических кружков любой возрастной категории, школьное учебное средство по курсам математики и ОИВТ, личное вспомогательное средство любителя и профессионала по курсу теории чисел. Заменяет собой таблицу простых чисел.

Входные данные: нечетные целые положительные числа в диапазоне от 5 до 10044017 включительно (10044017 — простое число).

В диапозоне от 10044019 до 29999873 включительно программа частично работоспособна, т.е. некоторые простые числа интерпретируются как составные и генератором пропускаются. Число 29999873 — последнее простое фиксируемое генератором. В диапазоне нечетных чисел от 29999875 до 99999999 генератор полностью неработоспособен, программа "бесконечно" зацикливается.

Длина программы: 24 шага. Количество клавшиных нажатий при записи программного текста и минимальное время этой записи в секундах — 46. Используется упрощенный алгоритм Эратосфена, в котором множество простых допустимых делителей в диапазоне от 3 до 3169 расширено до всех нечетных чисел этого основного диапазона. Переменные величины и используемые для них регистры: Д = сРг5 — текущий нечетный делитель, И = сРг6 — нечетное испытуемое число, ЧАСТН = И/Д = сРгХ1. Словесный псевдокод алгоритма: для нарастающих значений Д из допустимого диапазона, начиная с Д = 3, вычислять ЧАСТН = И/Д; в случае равенства И/Д и целой части от И/Д досрочно переходить на новое значение нарастающего нечетного И, выполняя И:И + 2; если Д больше ЧАСТН, считать И + П, где П — простое число и осуществлять стоп-индикацию результата с последующим переходом на счет с новым значением И. ЗАМЕЧАНИЕ К АЛГОРИТМУ: проверка по равенству целого от И/Д и исходного И/Д дает возможность несколько сократить текст программы и повысить ее быстродействие по сравнению с аналогами (см. ВТиП 4/1988, программу # 018 в статье "Алгоритмический букварь и карманная ЭВМ"), но при восьми значащих разрядах И и при конкретных его значениях, отмеченных выше, алгоритм (программа) частично или полностью теряет работоспособность.

Оценка быстродействия программы: от 11 к 11 — за 9 сек, от 101 к 101 — за 21 сек, от 10044017 к 10044017 — за 2 часа 12 мин 23 сек, от 29999873 к 29999873 — за 3.5 час.

Далее приводится текст программы:

00	Х-+6	46	/ + **Фиксация в Рг6 стартового испытуемого И**
01	1	01	/ + **Клавишная константа единица, или КЛАВКО 1**
02	Х-+П 5	45	/ + **Передача Д = сРгХ = 1 в Рг5, или Д: = 1, т.к. сРг5 = Д**
03	К П-+Х 5	Г5	/ + **Д: + Д + 1, начальная фаза генерации нового Д**
04	К П-+Х 5	Г5	/ + **Д: = Д + 1, конец генерации Д; новД = старД + 2**
05	П-+Х 6	66	/ + **Вызов из Рг6 текущего значения И**
06	П-+Х 5	65	/ + **сРг5 — на РгХ, или Д; сРгХ = И — РгУ**
07	-"-	13	/ + **И/Д; буквой К обозначена вспомог.переменн.**
08	К ЦЕЛ	34	/ + **Выделение целой части К от И/Д**
09	Ф Вх	0	/ + **Вызов на РгХ первоначального знач.И/Д из РгХ1**
10	—	11	/ + **М = К-И/Д; М — вспомогательная переменная**
11	Ф ХнрвнО	57	/ + **М не равно нулю?**
12	2 0	20	/ + **НЕТ, М равно нулю; переход на шаг 20**
13	Ф Вх	0	/ + **Да, М не равно нулю; вызов на РгХ ЧАСТН = И/Д**
14	П-+Х 5	65	/ + **Вызов на РгХ из Рг5 значения текущ. делит. Д**
15	—	11	/ + **ЧАСТН — Д = Т; ЧАСТН = И/Д, Т — вспом. переменн.**
16	Ф ХмншО	5С	/ + **Т меньше нуля?**
17	0 3	03	/ + **НЕТ, Т больше или равно нулю; ход на шаг 03**
18	П-+6	66	/ + **ДА, Т меньше нуля, Д больше ЧАСТН; вызов И = П**
19	С/П	50	/ + **Стоп-индикация простого числа П = И**
20	К П-+Х 6	Г6	/ + **И: = И + 1, начальная фаза генерации нового И**
21	К П-+Х 6	Г6	/ + **И: = И + 1, конец генерации; новИ = старИ + 2**
22	БП	51	/ + **Заголовок команды безусловного перехода**
23	0 1	01	/ + **Ход на шаг 01 к новой серии вычислений**

ИНСТРУКЦИЯ: В/О, НАБОР И, "И", С/П, "П", С/П, "П" и так далее

Соблюдать оговоренные границы диапазонов. При работе в пределах ограниченной работоспособности пользоваться вспомогательной таблицей "необнаруживаемых" простых чисел.

ПАПА Простейшая Авторская Прикладная Алгоритмика ПАПА

Программа Угадай число

Автор: Бударевский Андрей Станиславович. Комментарии: Тарасенко Борис Алексеевич. Авторский вариант разработан в 1990 году. Публикуется впервые. Прототипы есть. Реализуется на калькуляторах МК-61, МК-52.

Назначение: учебно-развлекательная программа, тренирующая навыки устного счета и логичность мышления. Области применения: кружки программирования и математики, школьные курсы математики и ОИВТ, обеспечение досуга любителя математики и программирования. Основное содержание программы: нужно угадать число, сформированное программой с помощью датчика случайных чисел, путем минимального, по возможности, количества попыток. Количество попыток программой не фиксируется.

Длина программы: 24 шага. Количество нажатий клавиш при записи программного текста и минимальное время этой записи в секундах — 42. Переменные величины и используемые для них регистры: сPr4 — константа 100, сPrД — константа 8, сPr0 — константа 41СЛО, ЧИСЛО, сPr2 — константа 60ЛСЕ, или БОЛЬШЕ, сPr1 — константа СЛ60, или СЛБО, или СЛАБО (МЕНЬШЕ), сPrС — константа 17, или П, или ПРАВИЛЬНО, или ПОБЕДА. Время одного цикла "отгадывания" — до 4 секунд (3 + 1)7 сPrА = отгад.числу.

00	П-+Х 0	60	/ + **Вызов из PrО сообщения ЧИСЛО для пуска ГСЧ**
01	С/П	50	/ + **Останов и индикация запроса о вводе числа**
02	ВСТЕК	ОЕ	/ + **Инициализирующее ГСЧ дробное число в PrУ**
03	К СЧ	3Л	/ + **Разовое псевдослучайное число из ГСЧ**
04	П-+Х 4	64	/ + **Вызов из Pr# на PrХ константы 100**
05	x	12	/ + **Выделение двух левых разрядов в целые**
06	К ЦУЛ	34	/ + **Целое 2-разрядное "загаданное" число**
07	Х-+П А	4-	/ + **Запоминаем для дальнейшего использования**
08	П-+Х 0	60	/ + **Сообщение "число", приглашение к вводу отг.**
09	С/П	50	/ + **Стоп-индикация ЧИСЛО и ввод отгадки**
10	П-+Х А	6-	/ + **Вызов из PrА на PrХ загаданного числа**
11	—	11	/ + **Разность между "загадкой" и "попыткой"**
12	Ф Х=О	5Е	/ + **Разность равна нулю?**
13	1 6	16	/ + **НЕТ, сPrХнрвнО, ход на шаг 16**
14	П-+Х С	6С	/ + **ДА, сPrХ+О и ответ верный, число отгадано**
15	С/П	50	/ + **Стоп-индикация сигнала ПОБЕДЫ, 17, вызванн.**
16	Ф ХсншО	5С	/ + **Разность меньше нуля?**
17	2 1	21	/ + **НЕТ, сPrХбрвнО, ход на шаг 21**
18	П-+Х 1	61	/ + **ДА, сPrХмншО, вызов из Pr1 на PrХ СЛБО**
19	С/П	50	/ + **Стоп-индикация сообщения СЛАБО, или МЕНЬШЕ**
20	К БП Д	8Г	/ + **Безусловный переход на шаг 08, т.к. сPrД = 8**
21	П-+Х 2	62	/ + **Вызов из Pr2 на PrХ сообщения БОЛЬШЕ**
22	С/П	50	/ + **Стоп-индикация сообщения БОЛЬШЕ**
23	К БП Д	8Г	/ + **Безусловный переход на шаг 08 по сPrД = 8**

ПОСТКОММЕНТАРИЙ: в программе "загадываются" двухразрядные числа не более 100.

ПОДГОТОВКА К РАБОТЕ: она сводится к ручному клавишному формированию констант и сообщений и к занесению их в соответствующие регистры: набрать 100, Х-+П 4, набрать 8, Х-+П Д, 1418801, ВСТЕК, 1414301, К ИЛИ, К ДРБ, ВП, 5, ВСТЕК, К ЦУЛ, Х-+П О, (введено сообщение ЧИСЛО), 160888, ВСТЕК, 160346, К ИЛИ, К ДРБ, ВП, 5, ВСТЕК, Х-+П 2, (введено сообщение БОЛЬШЕ), 188601, ВСТЕК, 143601, К ИЛИ, К ДРБ, ВП, 4, ВСТЕК, К ЦЕЛ, Х-+П 1 (введено сообщение СЛАБО, или МЕНЬШЕ), 17, Х-+П С, (введено сообщение 17, или П, или ПОБЕДА)

ОСНОВНАЯ ИНСТРУКЦИЯ К РАБОТЕ:

- В/О, С/П, "ЧИСЛО", ввод произвольного дробного числа, С/П, "ЧИСЛО", набор числа-попытки не более 100, С/П, "БОЛЬШЕ или МЕНЬШЕ", С/П, "ЧИСЛО", набор следующей попытки, С/П и так далее, пока не появится сообщение 17 (ПОБЕДА, или ПРАВИЛЬНО);
- для проведения нового цикла отгадывания нужно повторить полностью все ручные операции основной инструктивной части.

Нам пишут

Завершает этот выпуск триада программ для МК-85 (на языке БЕЙСИК), присланная нам из подмосковного поселка Болшево профессиональным программистом Забегаевой Т.Л. Приводим письмо, сопровождающее программы, практически полностью:

"Недавно мы купили микрокомпьютер МК-85. У нас в семье есть дети и, естественно, они не могли остаться в стороне от этого события.

Известно, что если дети младшего школьного возраста пользуются для вычислений микрокалькулятором, то у них не вырабатываются навыки учетного счета. А МК-85 может предоставить большие возможности как раз для приобретения и развития этих навыков. Хочу поделиться с вами моим, пока еще очень небольшим, опытом в этом направлении.

Сначала я составила программу для девятилетнего сына на повторение таблицы умножения. Эта программа выдает на индикатор пример на умножение или деление. Нужно ввести цифры ответа и нажать клавишу EXE. Если ответ правильный, на индикаторе высвечивается слово "Правильно" и после нажатия на клавишу EXE предлагается новый пример. Если же ответ был дан неверный, то высвечивается слово "Неправильно", а на индикаторе после нажатия на клавишу EXE повторяется этот же пример.

Для младшего сына, которому четыре с половиной года, я составила аналогичную программу на сложение и вычитание однозначных чисел. Сыну эта игра (а он воспринимает это занятие как игру) очень понравилась. За какие-то неделю-две он научился довольно уверенно оперировать числами в пределах 20.

Когда интерес у детей к игре с этими программами начал ослабевать, мы стали решать на скорость: ставилась задача сосчитать за одну минуту как можно больше примеров. Личный рекорд постепенно с развитием навыка устного счета возрастает, и это, в свою очередь, является стимулом к дальнейшим занятиям. Еще хочу предложить вашему вниманию игровую программу "Угадай число". Эта программа взята из книги "Домашний компьютер", авторы Р. Лоберг и Т. Лутц (М.: Детская литература, 1990), и интерпретирована мной для МК-85. За семь попыток нужно угадать число, лежащее в интервале от 0 до 100. На каждую попытку на индикаторе дается направляющий ответ "Много", "Мало" или, в случае правильного ответа, "Ты угадал".

Нахождение ответа в этой программе можно усложнить, предложив поиск числа в интервале от 0 до 1000. Для этого достаточно в операторе под меткой 20 функцию PAN# умножить не на 100, а на 1000. При этом целесообразно количество попыток увеличить до 10".

Советуем обратить на эти программы особое внимание, так как они оптимальны социально. При минимуме усилий, затрат оборудования и профессиональных навыков, получен стойкий и красивый результат. Все это естественно вписывается в рамки нашей темы и рубрики (см.текущие выпуски). Без высоких инстанций, без излишней навязчивости сей полезный опыт может быть перенесен во многие квартиры, где есть дети и уже есть МК-85.

Программа "Сосчитай"

Автор: Забегаева Т.А. Реализована на МК-85

Программа "СОСЧИТАЙ" предназначена для развития навыков устного счета у детей дошкольного и младшего школьного возраста. Программа предлагает примеры на сложение и вычитание чисел в пределах 20 и использует функцию генерации случайного числа RAN#, которая дает число, лежащее в интервале от 0 до 1. Умножив это число на 10 и выделив целую часть, получаем случайное однозначное число.

Примеры, предлагаемые программой "СОСЧИТАЙ", можно усложнить, распространив их на сложение и вычитание двузначных чисел. Для этого достаточно в операторах 10 и 15 функцию RAN# при определении переменных A и B умножать не на 10, а на 100.

```
5 PRINT "СОСЧИТАЙ"
10 A = INT(RAN#*10):B = INT(RAN#*10)
15 D = INT(RAN#*10)
20 C = A + B
30 IF D>4 THEN 90
40 PRINT A; " + "; B; " = ";
50 CSR 8:INPUT G
60 IF G = C THEN 80
70 PRINT "Неправильно": GOTO 40
80 PRINT "Правильно": GOTO 10
90 PRINT C; "-"; A; " = ";
100 CSR 9: INPUT G
110 IF G = B THEN 80
120 PRINT "Неправильно": GOTO 90
```

После запуска программы на индикаторе высвечивается случайный пример на сложение и вычитание. Требуется ввести цифры ответа и нажать клавишу EXE. Если ответ правильный, на индикаторе загорается слово "Правильно" и, после нажатия на клавишу EXE, предлагается но-

вый пример. Если ответ был дан неверный, то загорается слово "Неправильно", и на индикаторе повторяется этот же пример.

Количество переменных 5;

Количество шагов 178.

Программа "Таблица умножения"

Автор: Забегаева Т.А. Реализована на МК-85.

Программа "ТАБЛИЦА УМНОЖЕНИЯ" предназначена для повторения в игровой форме таблицы умножения детьми младшего школьного возраста. Программа предлагает примеры на умножение однозначных чисел и использует функцию генерации случайного числа RAN", которая дает число, лежащее в интервале от 0 до 1. Умножив это число на 10 и выделив целую часть, получаем случайное однозначеное число.

```
5 PRINT "ТАБЛИЦА УМНОЖЕНИЯ"
10 A = INT(RAN#*10): B = INT(RAN#*10)
15 D = INT(RAN#*10)
20 IF A = 0 THEN 140
30 C = A*B
40 IF D>4 THEN 100
50 PRINT A; "*";B;" = ";
60 CSR 6: INPUT G
70 IF G =  C THEN 90
80 PRINT "Неправильно"; GOTO 50
90 PRINT "Правильно" GOTO 10
100 PRINT C; ":"; A; " = ";
110 CSR 7: INPUT G
120 IF G = B THEN 90
130 PRINT "Неправильно": GOTO 100
140 A = 5: GOTO 30
```

После запуска программы на индикаторе высвечивается случайный пример из таблицы умножения. Требуется ввести цифры ответа и нажать клавишу EXE. Если ответ правильный, то на индикаторе загорается слово "Правильно" и, после нажатия на клавишу EXE, предлагается новый пример на умножение или деление. Если ответ был неверный, то загорается слово "Неправильно" и на индикаторе повторяется этот же пример.

Количество переменных 5;

Количество шагов 202.

Программа "Угадай число"

Автор: Забегаева Т.А. Реализована на МК-85.

Программа "УГАДАЙ ЧИСЛО" — игровая и представляет адаптированный для МК-85 вариант из книги Р.Лоберг, Т.Лутц "Домашний компьютер". За семь попыток, используя направляющие ответы, требуется угадать число, лежащее в интервале от 0 до 100.

```
10 PRINT "УГАДАЙ ЧИСЛО"
20 T = 0: N = INT(RAN#*100)
30 T = T  +  1
40 IF T = 8 THEN 120
50 PRINT "П"; T;
60 CSR 7: INPUT G
70 IF G = N THEN 110
80 IF G>N THEN 100
90 PRINT G; "Мало": GOTO 30
100 PRINT G; "Много": GOTO 30
110 PRINT "Ты угадал": GOTO 140
120 PRINT "Все попытки исчерпаны,";
130 PRINT "а число было"; N
140 PRINT "Сыграем еще?": GOTO 20
```

Количество переменных 3;

Количество шагов 234.

ИНВЕСТСЕРВИС

Вы хотите иметь надежный многотерминальный комплекс и у вас нет валюты — обращайтесь во Львовское МП "Инвестсервис"!



В сжатые сроки (максимум три недели) за РУБЛИ, по доступным ценам, с высоким качеством мы поставим, внедрим и адаптируем многотерминальный комплекс КРАБ.

Комплекс позволяет создать четыре дополнительных рабочих места, оснащенных персональной ЭВМ типа PC XT/AT.

Многотерминальный комплекс имеет программную поддержку, обеспечивающую одновременную работу нескольких пользователей в среде MS DOS.

Большую часть программного обеспечения, разработанного для PC XT/AT, можно эксплуатировать на комплексе без каких-либо доработок.

Многотерминальный комплекс особенно эффективен в приложениях, требующих ввода больших массивов данных при относительно небольшом объеме вычислений.

КРАБ

Применение многотерминального комплекса КРАБ на складах, в регистратурах поликлиник, библиотеках, учебных заведениях чрезвычайно выгодно, так как он заменяет несколько дорогостоящих компьютеров. Возможно подключение принтеров.

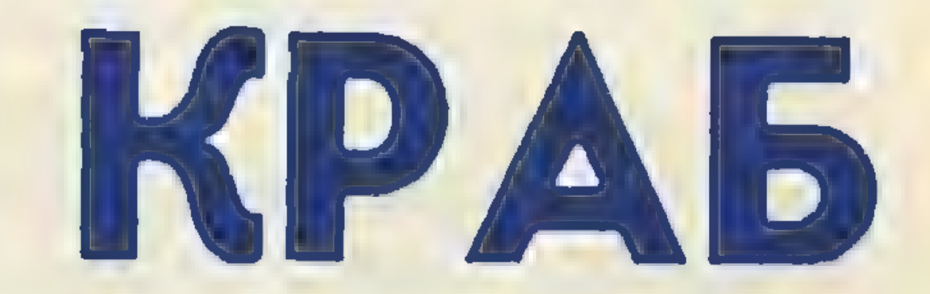
КОМПЛЕКТ ПОСТАВКИ:

- четыре терминала;
- плата расширения, устанавливаемая в ПЭВМ;
- кабели подключения;
- базовое программное обеспечение.

По желанию заказчика в комплект могут быть включены:

- импортная персональная ЭВМ;
- пакеты отладочных кросс-средств практически для любых микропроцессоров и однокристальных ЭВМ.

Обращаться по адресу:
290044, Львов-44, а/я 8863
МП "Инвестсервис"
Справки по телефонам:
35-35-79, 34-32-12 с 8 до 17 часов;
34-29-42 с 18 до 7 часов.



III Всесоюзная олимпиада по информатике

В Харькове состоялась III Всесоюзная олимпиада по информатике. Старшеклассники почти из всех республик (не была представлена только Литва) участвовали в двух турах олимпиады: теоретическом и практическом.

Решая задачи первого тура, они демонстрировали свое умение логически мыслить, проявлять смекалку и изобретательность, а также умение создавать математические модели и их анализировать. Второй тур — это экзамен на технику работы за пультом ЭВМ. Школьникам были предоставлены на выбор ПЭВМ "ЯМАХА" или IBM PC.

Из 87 участников лучшие результаты у Дмитрия Козлова (Ленинград), Юрия Зайцева (Украина) и Олега Таборовец (Белоруссия). Дипломами отмечены работы Рейнв Варблане (Эстония), Алексея Демакова (РСФСР).

Членам нашего клуба предлагаются две задачи теоретического тура. Представьте, что вы на олимпиаде и дерзайте, попробуйте решить задачи за 6 часов.

1. Круг разрезан по несамопересекающейся ломаной с вершинами, каждая из которых задана парой натуральных чисел (X1,У1,...,Хк,Ук).

Первая и последняя вершины лежат на границе круга, а остальные внутри его. Определить, можно ли раздвинуть круг по линии разреза на две части (выход из плоскости круга не допускается)?

2. Сколько (P, Q) коней необходимо для контроля:

а) ограниченной шахматной доски размером MxK клеток;

б) бесконечной полосы шириной M клеток;

с) неограниченной плоскости.

Примечание 1: (P, Q) конь — шахматная фигура, которая за один ход перемещается на P клеток по горизонтали и Q клеток по вертикали или на P клеток по вертикали и Q клеток по горизонтали. Например, (1, 2) — это обычный шахматный конь.

Примечание 2: Фигура или группа фигур контролирует поле, если каждая клетка поля доступна за один или несколько ходов хотя бы одной фигуре. Например, для контроля доски размером 8x8 необходим один конь или два слона.

В.Н.Касаткин

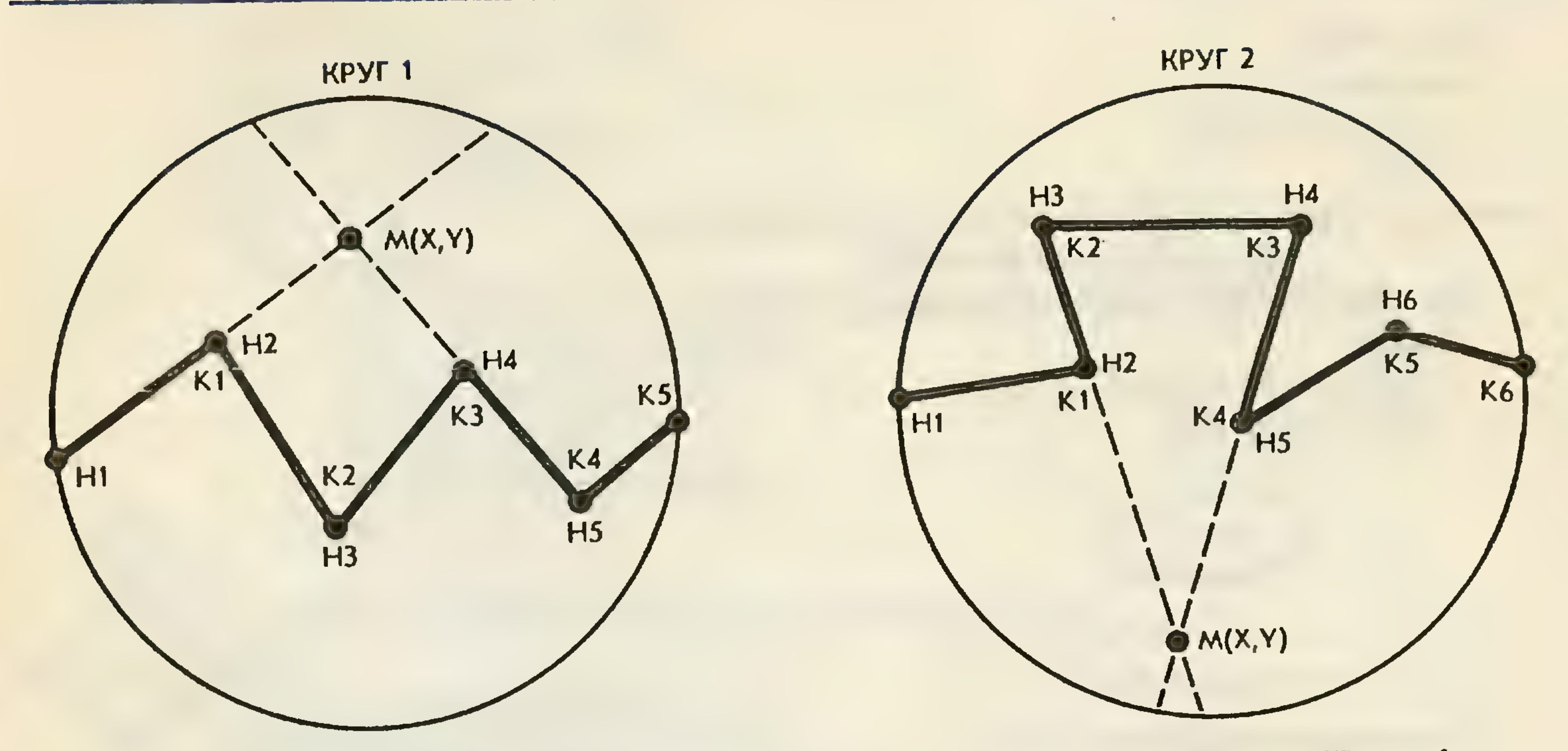
Задача Всесоюзной олимпиады по основам информатики 1990 г.

Юрий Волынский
(г.Симферополь, МАН "Искатель")

Краткое описание алгоритма

Рассмотрим круги 1 и 2, изображенные на рисунках. В каждом из них проведены ломаные, разбивающие круги на две части. Требуется определить, раздвинутся ли части кругов на плоскости.

Для ответа на вопрос будем перебирать все пары звеньев ломаной, исключая соседние. Перебор осуществляется следующим образом: первое звено (i) рассматривается со всеми последующими звеньями (к), начиная с третьего; второе (i) — со всеми последующими (к), начиная с четвертого и т.д., при этом звенья (i) назовем базовыми , а звенья (к) — сравниваемыми. Для каждого звена определим "начало" и "конец" так, что первая вершина ломаной будет началом первого звена; вторая — концом первого и началом



второго; третья — концом второго и началом третьего и т.д. Перебор осуществляется до тех пор, пока не встретится пара, для которой выполняется нижеследующее условие, состоящее из двух частей:

1-я часть — начало i-го звена ближе к точке пересечения прямых, образуемых рассматриваемыми звеньями, чем его конец;

2-я часть — конец к-го звена ближе к точке пересечения прямых, образуемых рассматриваемыми звеньями, чем его начало. Тогда перебор прекращается и можно сказать, что части круга не раздвинутся. Если такой пары не встретится, то части круга раздвинутся.

Если точка пересечения прямых принадлежит i-му звену, то достаточно проверить выполнение только второй части условия, если же она принадлежит второму звену, то достаточно проверить выполнение только первой части условия.

Если два рассматриваемых звена имеют общую точку, то части круга не раздвинутся. Параллельные звенья не рассматриваются.

YY,Z1,Z2 — булевы переменные;
YY = {истинна, если полуплоскости раздвигаются, иначе ложна;
N — количество вершин ломаной;
I — обозначает базовые звенья ломанной;
K — обозначает сравниваемые с базовыми звенья ломаной;
N(I),N(I + 1) — вершины, образующие базовое звено;
N(K),N(K + 1) — вершины, образующие сравниваемое звено;
Z(1) — базовое звено;
Z(2) — сравниваемое с базовым звено;
M(X,Y) — точка пересечения прямых, образуемых звеньями Z(I),Z(K), с координатами X,Y;
N1 — расстояние от N(I) до M(X,Y);
K1 — расстояние от N(I + 1) до M(X,Y);
N2 — расстояние от N(K) до M(X,Y);
K2 — расстояние от N(K + 1) до M(X,Y);

```
PROGRAM {СДВИГ НА ПЛОСКОСТИ} A (input,output);
    type massiv = array[1..60] of real;
    var i,k,kol_ver:integer;
        nsn_ysl,z1,z2:boolean;
        xm,ym,n1,k1,n2,k2,f:real;
        x,y:massiv;
    procedure swapval (a,b:real);
```

```
    var k:real;
     begin k: = a;
        a: = b;b: = k
     end;
function t_per (x1i,x2i,y1i,y2i,x1k,x2k,y1k,y2k:real;
               var xx,yy:real):boolean;
    var dx1,dx2,dy1,dy2,c1,c2,dx,dy,d;real;
     begin
       dx1: = x2i-x1i;dx1: = -dx1;
       dy1: = y2i-y1i;
```

НАЧАЛО
Ввод координат вершин
I: = 0
4
I: = I + 1
K: = I + 1
3
K: = K + 1
YY: = faLse
Z1: = faLse
Z2: = faLse
Вычислить M(X,Y)
т.M существует
–
+
YY: = true
MeZ(I)
+
–
Z1: = true
NeZ(K)
Z2: = true
1
2
2
Z1 and Z2
–
+
Z1
–
+
Вычислить N1, K1
N1 < K1
–
+
YY: = true
Z2
–
+
Вычислить N2, K2
K2 < N2
–
+
YY: = true
1
3
K ≥ N-1 or $\overline{YY}$
–
+
4
I ≥ N-3 or $\overline{YY}$
–
+
YY
+
–
Раздвигается
Не раздвигается
КОНЕЦ

```
          dx2: = x2k-x1k;dx2: = -dx2;
          dy2: = y2k-y1k;
          d: = dy1*dx2-dy2*dx1;
          t_per : = (d<>0);
          if d <> 0
             then begin
                    c1: = dy1*x1i + dx1*y1i;
                    c2: = dy2*x1k + dx2*y1k;
                    dx: = c1*dx2-c2*dx1;
                    dy:dy1*c2-dy2*r1;
                    xx: = dx/d;yy: = dy/d;
                  end;
        end;
  function prinad_zveny(a1,a2,b1,b2,xx,yy:real):boolean;
        begin
           if a1>a2 then swapval(a1,a2);
           if b1>h2 then swapval(h1,h2);
           prinad_zveny: = ( (xx> = a1) and (xx< = a2)
                              and (yy> = b1) and (yy< = b2) )
        end;
  function rasst (x1,x2,y1,y2:real):real;
        begin
           rasst: = sqr((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1))
        end;


BEGIN
  write('Количество вершин ломаной ? ');readln( kol_ver);
  if kol_ver > 3 then
 Begin
  writeln('');
  writeln('Ввод координат вершин. ');
  for i: = 1 to kol_ver do
    begin
       writeln('= = = = = = =',i,'-ая вершина = = = = = = = ');
       write('Координата x? ');read(x[i]);
       write('Координата Y? ');read(y[i]);
    end;
 writeln;
   i: = 0;
 repeat i: = i + 1;
          k: = i + 1;xm: = 0;ym: = 0;
   repeat k: = k + 1;
       osn_ysl: = false; z1: = false; z2: = false;
       if t_per( x[i],x[i + 1},y[i],y[i + 1];
                 x[k],x[k + 1],y[k],y[k + 1],xm,ym)
   then begin
          if prinad_zveny(x[i],x[i + 1],y[i],y[i + 1],xm,ym)
            then z1: = true;
         if prinad_zveny(x[k],x[k + 1],y[k],y[k + 1],xm,ym)
            then z2: = true;
          if not ( z1 and z2 )
            then begin
                   if not(z1)
                     then begin
```

```
                    n1: = rasst(xm,x[i],ym,y[i];
                    k1: = rasst(xm,x[i + 1],ym,y[i + 1]);
                    if not ( n1<k1 ) then osn_ysl: = true;
                    end;
              if not(z2)
                then begin
                    n2: = rasst([m,x[k],ym,y[k];
                    k2: = rasst(xm,x[k + 1],ym,y[k + 1]);
                    if not  (k2<n2 ) then osn_ysl: = true;
                    end;
               end;
            end
          else osn_ysl: =  true;
    until ( (k> = kol_ver-1) or not (osn_ysl) );
  until ( (i> = kol_ver-3) or not (osn_ysl) );
  if osn_ysl
    then writeln('РАЗДВИГАЕТСЯ')
    else writeln('НЕ РАЗДВИГАЕТСЯ');
 End
    else writeln('РАЗДВИГАЕТСЯ');
 readln(f);
END.
```

Андрей Дымура (Кировоград)

ИГРА ГРАНДИ

Успех в использовании ЭВМ

Условия игры

Имеется одна общая группа, содержащая n предметов. Играют двое. Ходят поочередно. Игра начинается с того, что один из соперников разбивает исходную гр /ппу на две неравные. Далее, выполняя очередной ход, каждый соперник разбивает любую из имеющихся групп на две неравные.

Игра продолжается до тех пор, пока все группы не будут состоять из одного или двух предметов. Победитель в игре считается тот, кто сможет выполнить последнее разбиение.

Меня очень заинтересовала возможность построения общего алгоритма для этой игры. Хочу сообщить, что алгоритм беспроигрышного поведения в игре Гранди уже найден. Этот алгоритм обладает одной интересной особенностью: он годится для любого заданного числа n, однако для этого же n должна быть построена специальная вспомогательная таблица. Я попытался построить таблицу для n = 50. Это оказалось принципиально не сложным, но довольно трудоемким занятием. Причем с ростом n быстро растет и трудоемкость заполнения таблицы. Здесь уже не обойтись без компьютера.

Автором вышеупомянутого алгоритма беспроигрышного поведения в игре и правил формирования вспомогательной таблицы является мой учитель программирования, преподаватель Кировоградского высшего летного училища гражданской авиации Кузнецов С.Т.

Привожу ниже, с разрешения Сергея Тихоновича, идею алгоритма и программу на БЕЙСИКе для формирования вспомогательной таблицы.

Суть алгоритма

Алгоритмы, задающие беспроигрышную стратегию в играх, должны содержать ответ на два вопроса:

- Каким по очереди вступать в игру?
- Как расчитывать каждый свой ход?

Ответы на оба вопроса для игры Гранди можно всегда получить на основании информации, содержащейся во вспомогательной таблице. Здесь представлена часть таблицы, которой можно пользоваться, если число предметов в каждой из имеющихся групп не превосходит пятидесяти четырех.

Зная начальное количество предметов n из таблицы, сразу же можно узнать, каким по очереди вступать в игру, и если вступать первым, то какой ход нужно сделать. На протяжении всей игры, из той же таблицы мы сможем получать рекомендации относительно каждого своего хода. В итоге мы всегда одержим победу, выполнив последнее разбиение.

Весь процесс игры можно рассматривать как смену проигрышных и выигрышных позиций для каждого из соперников. Известно, что всякую выигрышную позицию (для делающего ход) можно всегда перевести в проигрышную позицию для соперника. Однако свою проигрышную позицию никаким ходом нельзя перевести в проигрышную для соперника.

Цель первого, как и всех последующих ходов, — создание проигрышной позиции для соперника.

Для делающего ход признаком проигрышной позиции является равенство нулю **ним**-значения. Величина **ним**-значения выбирается из второго столбца таблицы.

Ним-значение есть двоичная функция от n (NZ(n)). Если NZ(n) = 0, то первый ход следует предложить сделать сопернику. Если же NZ(n) = 0, то в игру следует вступать первым.

Рассмотрим на примере расчет своего первого хода. Пусть n = 48. Соответствующее $NZ(48) = 100_2 = 0$. Это значит, что первый ход за нами. Количество отделяемых предметов находим в нулевом столбце вторых **ним**-сумм (NS2 = 0). В нашем примере на пересечении строки n = 48 и столбца NS2 = 0 обнаруживаем число 21 — это и есть отделяемая нами на первом ходе группа предметов.

Рассмотрим пример расчета очередного хода. Пусть в процессе игры образовались шесть групп, содержащих 27,28,35,47,30 и 14 предметов.

Соответствующие им **ним**-значения находим из таблицы:

$$NZ(27) = 100_2, \quad NZ(47) = 101_2$$
$$NZ(28) = 001_2, \quad NZ(30) = 011_2$$
$$NZ(35) = 010_2, \quad NZ(14) = 010_2$$

В дальнейшем будем использовать понятие первой ним-суммы. Первая ним-сумма есть сумма по модулю два всех **ним**-значений. В данном случае имеем:

$$\begin{array}{rcl} 27 & — & 100 \\ 28 & — & 001 \\ 35 & — & 010 \\ 47 & — & 101 \\ 30 & — & 011 \\ 14 & — & \underline{010} \\ & & 011 \quad NS1 = 011_2 \end{array}$$

Полученное значение $NS1 = 011_2$ позволяет определить, какую из шести групп разбивать и как это делать.

Вспомогательная таблица

Число предметов, n	Ним-значение NZ(n)	Вторая ним-сумма NS2				
		000	001	010	011	100
1	000					
2	000					
3	001	1				
4	000					
5	010	1	2			
6	001	1				
7	000					
8	010	1	2			
9	001	2				
10	000					
11	010	1	2			
12	001	2				
13	011	5	1	2		
14	010	4	2			
15	001	3				
16	011	5	1	2		
17	010	7	2			
18	100	3	5	1	2	
19	011	5	4	2		
20	000					
21	100	1	5	4	2	
22	011	2	7	3		
23	000					
24	100	1	5	7	2	
25	011	2	10	3		
26	000					
27	100	1	5	10	2	

Число предметов, n	Ним-значение NZ(n)	Вторая ним-сумма NS2				
		000	001	010	011	100
28	001	2				
29	010	13	1			
30	011	4	2	1		
31	001	3				
32	010	13	1			
33	100	7	2	1	5	
34	001	3				
35	010	13	1			
36	100	10	2	1	5	
37	001	3				
38	010	13	1			
39	100	18	2	1	5	
40	001	3				
41	101	16	1	13	3	2
42	100	18	2	4	5	
43	001	3				
44	101	19	1	13	6	2
45	100	18	2	7	5	
46	001	3				
47	101	22	1	13	9	2
48	100	21	2	10	5	
49	001	3				
50	000					
51	010	1	2			
52	001	2				
53	000					
54	010	1	2			

Находим старший разряд в NS1, который содержит "1" — это в нашем примере второй разряд. Отсюда следует рекомендация: разбиению следует подвергать ту из групп, в записи **ним**-значения которой имеется единица во втором разряде.

В данном случае разбиению можно подвергать любую из трех групп в 35,30 или 14 предметов.

Пусть выбрана группа, содержащая 14 предметов. Для этого находим вторую **ним**-сумму: NS2 = NS1 + NZ (14). Вторая **ним**-сумма (NS2), как и первая (NS1), есть сумма чисел по модулю два. В нашем случае $NS2 = 001_2 + 010_2 = 001_2$. Теперь, зная $NS2 = 001_2$, количество отделяемых предметов находим на пересечении строки n = 14 и столбца $NS2 = 001_2$. Там обнаруживаем число "2". Следовательно, в резальтате нашего хода группа из 14 предметов распадается на две: 2 и 12.

После этого в игре образуется семь групп и соответствующая $NS1 = 000_2$. Следовательно, как и предполагалось, соперник вынужден делать ход в проигрышной для него позиции.

Из сказанного ясно, что главным является формирование вспомогательной таблицы.

Ниже приводится текст программы, которая генерирует вспомогательную таблицу для любого наперед заданного натурального n. Естественно, ограничения на n налагаются физическими возможностями конкретного компьютера.

Программа-генератор вспомогательной таблицы

```
10 CLS
20 INPUT "Число предметов n"; V%
30 DIM T% (63),K%(V%)
40 N% = 0
50 N% = N% + 1
60 IF N% > V% THEN END
70 J% = -1
80 J% = J% + 1
90 IF T%(J%) > 0 THEN 80
100 K%(N%) = J%
110 LPRINT "n = ";N%;"NZ = ";BIN$(K%(N%));"        NS2:";
120 I = 0
130 IF I% = J% THEN 170
140 LPRINT T%(I%);
150 I% = I% + 1
160 GOTO 130
170 LPRINT
180 FOR I% = 0 TO 63
190 T%(I%) = 0
200 NEXT I%
210 M% = INT(N%/2)
220 I% = M%
230 IF I% = 0 THEN 50
240 S% = K%(I%) XOR K%(N% + 1-I%)
250 T%(S%) = I%
260 I% = I%-1
270 GOTO 230
```

После введения пользователем числа предметов программа на принтер выводит компоненты таблицы.

Краткость программы позволяет надеяться, что заинтересованный читатель по ее тексту сможет восстановить для себя алгоритм формирования таблицы.

Отмечу кратко, что мне удалось составить программу-консультант для игры Гранди, которая для определенных групп предметов, возникших на любом этапе игры, либо сообщает выигрышный ход, либо выдает информацию о проигрышности исходной позиции. Программа работает с готовым фрагментом вспомогательной таблицы, и поэтому в моем варианте число предметов в каждой группе не может быть более ста.

Литература

1. *Гарднер М.* Математические головоломки и развлечения. — М.: Мир, 1971.
2. *Гарднер М.* Математические новеллы. — М.: Мир, 1974.
3. *Касаткин В.Н., Владыкина Л.И.* Алгоритмы и игры. Киев: Рад.шк., 1984.
4. *Нильсон Н.* Искусственный интеллект. — М.: Мир, 1973.

Комментарий специалиста

Сообщение кировоградского десятиклассника Андрея Дымуры, пришедшее недавно в клуб "Терминал", вызывает большой интерес.

Секрет беспроигрышного поведения в игре Гранди непрост. Немало математиков во всем мире трудились над его поиском.

Предложенный выигрышный алгоритм, безусловно, красив. Но важнее, пожалуй, следующее обстоятельство. Существует общеизвестная, хорошо разработанная еще в 1901 году профессором Гарвардского университета Чарльзом Л.Бутоном теория игры НИМ. Именно обобщение этой теории привело к построению алгоритма игры Гранди. Дело в том, что в игре НИМ ним-значение (NZ(n)) совпадает с двоичной записью числа n, а вторая ним-сумма (NS2), переведенная в десятичную систему, совпадает с числом предметов, которое нужно оставить в соответствующей группе. Эти совпадения можно считать "счастливой случайностью". В других подобных играх дело обстоит не так просто, и тогда, как в игре Гранди, нужно пользоваться вспомогательной таблицей.

Я располагаю любопытными примерами игр, которые сообщил мне С.Кузнецов. Они порождены изменением правил игр НИМ и Гранди. Игры на первый взгляд разные. Однако небольшие изменения в программе-генераторе вспомогательной таблицы позволяют конструировать для этих игр выигрышные алгоритмы.

Вышеизложенное позволяет говорить не об отдельных играх, а о целом классе игр, которые условно можно назвать "Ним-игры".

В.Н.Касаткин

Задачник клуба "Терминал"

Ниже предлагается задача для разработки алгоритма и программы ее решения на ЭВМ. Можно предлагать только алгоритм или алгоритм и программу. Одну программу присылать не следует.

Лучшие решения задач будут опубликованы и прокомментированы специалистами.

Известна задача-шутка "Как из мухи сделать слона? Речь идет о поиске цепочки из слов (имен существительных, нарицательных, в единственном числе, именительном падеже), первое из которых МУХА, последнее СЛОН, а все промежуточные отличаются одной буквой и встречаются в цепочке один раз.

Попробуйте найти такую цепочку из слов.

Рассказанное позволяет сформулировать задачу на программирование: пусть дан алфавит А из нескольких слов. Считая одно из слов A_1 начальным, а другое A_k конечным, построить цепочку слов от начального к конечному. В цепочку могут входить только слова из данного алфавита А, не более, чем по одному разу. Слова-соседи по цепочке должны отличаться только одной буквой.

Прилагаем далее одно из возможных решений этой задачи.

В.Касаткин

Алексей Юшин
(МАН "Искатель", г.Симферополь)

Программа "Слово за-слово"

Известна задача-игра, в которой из заданного слова требуется вывести другое. При этом разрешается использовать только существительные и на каждом шаге менять лишь одну букву текущего слова.

Назовем два слова равной длины соседями, если они отличаются только одной буквой. Тогда задачу можно сформулировать так.

Дан словарь V и слова s и t, принадлежащие V.

Получить последовательность

$R = (s, r_1, r_2, \dots, r_n, t)$,

все смежные элементы которой соседи, r_i R.

Запишем на ПАСКАЛЕ функцию булевского типа Neighbour (a,b), где a и b слова, принадлежащие V. Значение функции равно TRUE, если a и b соседи, и FALSE в противном случае.

```
FUNCTION Neighbour (a,b : string): Boolean;
   Var
      i : Range ;
      Y : Boolean ;
      c,d : String ;
   begin
      Y : = (Length(a) = Length(b)) and (a<>b) ;
         if Y
            then
               begin
                  i : = 1 ;
                  REPEAT
                    c := a ;
                    d := b ;
                    c[i] := "*" ;
                    d[i] := "*" ;
                    Y := (c = d) ;
                    i := i + 1 ;
                  UNTIL Y or (i > Length(a)) ;
               end ;
      Neighbour := Y;
   end;
```

Алгоритм построения цепочки R можно сформулировать так:

1. Включить слово — источник S в R;

2. ЕСЛИ последующее слово в R есть цель t, то напечатать цепочку R и цель t ;

ИНАЧЕ найти список Q, всех соседей последнего слова из R, вычеркнуть их из словаря V ;

3. ЕСЛИ список Q соседей последнего слова пуст, то удалить из R последнее слово и восстановить его в V ;

ИНАЧЕ взять первое слово из списка Q, вычеркнуть его из Q и включить в качестве последующего слова R;

4. Завершить алгоритм, если исчерпаны все списки соседей.

Таким образом, в ходе алгоритма изменяется состояние словаря V за счет удаления и восстановления в нем слов. Каждому слову цепочки ставится в соответствие список Q его соседей. Каждый такой список в ходе алгоритма исчерпывается удалением его начальных элементов.

Этот алгоритм реализуем в виде рекурсивной процедуры, с помощью которой можно получить все цепочки выводящие t из V.

Ниже приведен текст рекурсивной процедуры Resol. Перед обращением к ней следует исключить исходное слово из словаря V, чтобы предотвратить зацикливание. Для исключения слов из V используется булев вектор Vec. Если требуется исключить слово с номером i из V, то Vec[i] присваивается значение FALSE, при необходимости восстановления слова — значение TRUE. В процедуре Resol используется процедура Write_Path, которая осуществляет вывод на экран полученных цепочек слов.

```
PROCEDURE Resol (Word  : string       ;
                 Vec   : BoolArray    ;
                 Ind   : NatArray     ;
                 Numb  : Natural  ;
                 Path  : StringArray) ;
Var
        cnt,i : Natural ;
begin

        if Word  =  Target

            then Write_Path(Numb, Path)

            else

                begin

                    Numb           : =  Numb  +  1;
                    Path[Numb]     : =  Word        ;
                    cnt            : =  0         ;

                    for i : =  1 to NVoc do

                        If Vec[i] and Neighbour (Word,V[i])

                            then

                                Begin

                                    Vec [i] : =  False;

                                    Cnt : =  Cnt  +  1;

                                    Ind [cnt] : =  i;

                                End;

                    for i : =  1 to Cnt do

                        Resol (V[ind[i]],Vec,Ind,Numb,Path);

                end;

end;{Resol}
```

Комментарий специалиста

Рассмотренный алгоритм и программа могут служить элементарным введением в программирование логического вывода на основе знаний, заданных с помощью отношений между объектами некоторой предметной области. В задаче "Слово За-Слово" предметная область — это просто словарь (множество слов-существительных русского или английского языков).

Над множеством слов словаря задано отношение соседства, а логический вывод есть такая последовательность слов, в которой каждые два подряд записанных слова находятся в отношении соседства.

По мере того как строится логический вывод, словарь V усекается — в нем отмечаются как несуществующие те слова, включение которых в оставшуюся часть цепочки вывода запрещается.

Какие же элементы запрещается включать в логический вывод в процессе его построения?

Во-первых, это те слова, которые уже включены в цепочку.

Во-вторых, и это очень важно, в цепочку слов запрещается включать так называемых "ближайших соседей" только что внесенного слова. Ближайшие соседи — это слова, отличающиеся одной буквой в одной и той же позиции. Иными словами, процедура RESOL не включает в вывод подцепочки вида:

... ноль —> соль —> роль —>моль —>...

не приближающие нас к цели.

В то же время слова ноль, соль, роль и моль будут рассматриваться процедурой RESOL как альтернативные при построении различных цепочек. Это достигается тем, что в качестве недоступных отмечаются все соседи слова, только что включенного в цепочку вывода.

Таким образом, если

R_i
R_{i+1} соседи

R_{i+2} соседи

три последовательных слова в цепочке вывода, то R_{i+2} не может быть ни соседом R_i, ни ближайшим соседом R_{i+1}. В то же время R_{i+1} сосед R_i ; R_{i+2} сосед R_{i+1}

Процедура RESOL находит все цепочки вывода. Этому процессу соответствует дерево, в котором слово, приписанное каждой вершине (кроме корня) связано с потомками отношением соседства, но не ближайшего соседства.

Все потомки одного предка связаны отношением соседства, в том числе и ближайшего соседства.

Если в дереве вывода существуют концевые вершины, помеченные целевым словом t, то выводы — это последовательности слов, встречающихся на каждом пути от корня к таким концевым вершинам.

Дерево вывода является динамическим. Если процесс построения цепочки вывода (1) достиг цели или (2) зашел в тупик (текущее слово не имеет соседей), то осуществляется возврат на более высокий уровень. При этом концевые вершины удаляются из дерева, а из цепочки вывода возвращаются в словарь V, т.е. снова становятся доступными, слова, соответствующие этим концевым вершинам.

Этот достаточно сложный процесс с использованием рекурсии программируется очень лаконично.

Без сомнения, это главное достоинство предлагаемой программы.

Теперь вообразите, что слова в V — это формулы или операторы какого-либо языка программирования, а логические выводы — формульные вычисления или программы. Представьте себе также, что слова перед включением их в словарь V преобразуются по некоторым правилам. Вы окажетесь в увлекательном мире логических преобразований с помощью ЭВМ.

Задачи

1. Напишите булеву функцию, распознающую отношение ближайшего соседства.
2. Измените процедуру Resol на Opt_Resol, которая находит кратчайшую цепочку вывода.
3. Напишите программу визуализации дерева, соответствующего процессу логического вывода.

И.А.Переход

X 11 Hand held (О самых малых). — М.: Знание, 1991. — 48 с. — (Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение"; № 2).

ISBN 5-07-001715-2

35 к.

Hand Held — устройство, удерживаемое одной рукой, можно перевести иначе — ручной счет, или так... Во всяком случае речь в этом выпуске — о калькуляторе, о его работе и работе с ним. Все, что останется за рамками этой брошюры, будет продолжено в специальной рубрике в последующих выпусках.

Материал рассчитан на широкий круг читателей.

2302030000 ББК 32.85



Научно-популярное издание

HAND HELD

(О самых малых)

Гл.отраслевой редактор *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл.редактор *Н. А. Васильева*
Художник *В. Н. Конюхов*
Худож.редактор *И. А. Емельянова*
Техн.редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ № 11628

Подписано к печати 21.01.91. Формат бумаги 70x1001/16. Бумага офсетная. Печать офсетная. Усл.печ.л.3,90. Усл.кр.-отт.8,45. Уч.-изд.л.4,51. Тираж 48488 экз. Заказ 1920. Цена 35 коп. Издательство "Знание". 101835, ГСП, Москва, Центр, проезд Серова,д.4. Индекс заказа 914702. Отпечатано с оригинал-макета издательства "Знание" на ордена Трудового Красного Знамени Тверском полиграфическом комбинате Государственного комитета СССР по печати. 170024, г.Тверь, пр.Ленина,5.

Цена 35 коп.

Индекс 70195

Адрес подписчика:

Солн 5-27



Подписная
научно-
популярная
серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

И ЕЁ ПРИМЕНЕНИЕ

Наш адрес:
101835,
Москва,
Центр,
проезд
Серова,4